USENIX Proceedings of the 3rd Large Installation System Administration of Windows NT/2000 Conference *Seattle, Washington, USA* *August 2000*

conference

proceedings

3rd Large Installation System Administration of Windows NT/2000 Conference

Seattle, Washington, USA
August 1–2, 2000

Co-Sponsored by

USENIX
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

SAGE
THE SYSTEM ADMINISTRATORS GUILD

For additional copies of these proceedings contact:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710 USA
Phone: 510 528 8649
FAX: 510 548 5738
Email: office@usenix.org
WWW URL: http://www.usenix.org

The price is $18 for members and $24 for nonmembers.
Outside the U.S.A. and Canada, please add
$10 per copy for postage (via air printed matter).

ISBN 1-880446-19-7

Printed in the United States of America on 50% recycled paper, 10-15% post consumer waste.

USENIX Association

Proceedings of the

3rd Large Installation System Administration

# of Windows NT/2000 Conference

August 1–2, 2000
Seattle, Washington, USA

Conference Organizers

3rd Large Installation System Administration of Windows NT/2000 Conference

August 1–2, 2000
Seattle, Washington, USA

Message from the Conference Chairs

Welcome to Seattle and USENIX's 3rd Large Installation System Administration of Windows NT/2000 Conference.

This, the third installment of LISA-NT, follows the previous two in providing you with a unique forum for information exchange on administering large installations running Windows. As we all know, a large-scale environment presents a unique set of challenges. This year, with the release of Windows 2000, marks a special opportunity for further integration with the existing operating systems in our organizations, as the program of this conference shows.

The program committee reviewed twenty full-paper submissions. Each paper was reviewed by all members of the program committee. We met in March and selected nine papers for the program. Eight of the nine papers selected were from industry, while the ninth was an academic paper. We would like to thank the authors for their energy and generosity, and the program committee for contributing their time and knowledge to making this a better conference.

We are also pleased to include two invited talks in the conference. The presenters of the first, on securing Windows 2000, are two people who have insider knowledge: David LeBlanc and Mike Howard, both of Microsoft Corporation. The second talk is on Active Directory, given by Stuart Kwan, who brings with him some unique insights on Active Directory from Microsoft.

We cannot forget the opening and closing keynotes. The program committee had a difficult time deciding between two potential keynote speakers, so they decided to include both. The opening keynote speaker, Marcus Ranum of Network Flight Recorder, offers insights on the times we live in. William Hugh Murray, a security consultant to Deloitte & Touche, closes the conference with everything the security administrator needs to know about security.

We would like to acknowledge the efforts of key individuals who supported us in this project. The USENIX staff has been invaluable, and without them this conference would never have happened. Thank you to Judy DesHarnais, Ellie Young, Monica Ortiz, Jane-Ellen Long, and anyone else we might have forgotten.

Phil Cox, SystemExperts Corp.
Jesper M. Johansson, Boston University
Conference Co-chairs

Providing Secure Access to Information Using the Internet

John Holmwood, *TransCanada*
Kevin Reichert, *Alterna*
Blaine Feniak, *TransCanada*

Introduction

Historically, TransCanadaPipeLines Limited, like most companies, has had a security policy that required its computing infrastructure (networks and computing systems) to be isolated from public networks, including the Internet. The existing infrastructure took advantage of this isolation when it was designed. New business opportunities require TransCanada to open up its computing infrastructure to public networks, particularly the Internet. The security policies have been revised to ensure that, while everyone who has a need to access TransCanada information can do so, individuals who do not have a legitimate reason to see TransCanada information cannot do so.

Security Requirements

There are a few main security requirements for connecting TransCanada's computing infrastructure to public networks. First, the connection must not provide an unintended access capability into our infrastructure. Secondly, we must ensure that no one can access TransCanada's information passing through the public network, without permission. Finally, it must be possible to ensure that anyone, who accesses TransCanada's general computing infrastructure from outside that infrastructure, whether using public or private networks, is properly authenticated. This paper describes the analysis TransCanada undertook to determine its strategy and architecture for providing secure access to company information over the Internet.

What is a VPN

A Virtual Private Network (VPN) is composed of two fundamental components:

- A tunnel –a network of virtual circuits for carrying private traffic [1]over the Internet that is created by encapsulating the data in special IP packets. The virtual in VPN.

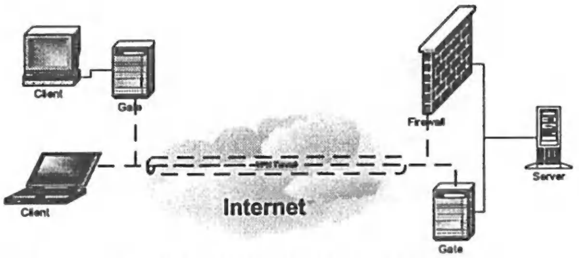


Figure 1 Possible Virtual Private Network Configurations

Figure 1 above depicts possible VPN configurations. A tunnel is set up from either an individualworkstation, or from a gate on a subnet at a remote site to a gate or firewall at the local site. There are a number of protocols that can be used to encapsulate the data in special IP packets, thereby creating the tunnel. See the section on protocols below.

- Security Services – the authentication, access control, confidentiality and integrity services. The private in VPN. These services are provided by cryptographic procedures such as encryption.

How to choose the right VPN

There are a very large number of products on the market that purport to be VPNs. For example, at the 1999 Interop+Networld conference in Las Vegas there were over 50 vendors who had VPN products based on IPSec. Therefore, it is very important to understand the features that distinguish these products, and to determine which of these features are needed for a particular application. The main distinguishing characteristics are:

- Firewall or Stand-Alone device
- Security services included
- Protocols supported
- Cost

Firewall/Stand Alone

Typically a VPN requires a device at the perimeter of the company network on which the tunnel terminates. Most VPN products can be divided into either products with tunnels that terminate on firewalls, or products with tunnels that terminate on stand alone devices (e.g. routers or servers).

Firewall

The products which terminate on firewalls generally come from firewall vendors Quel Surpris☺). They are either integral to the firewall product, or come as add on modules.

Advantages

The main advantage offered by firewalls that act as terminators for VPN tunnels is that only one device needs to be exposed to the Internet. This lowers the exposure to attack.

Concerns

A firewall that has to support a VPN tunnel as well as do its normal job of controlling access to the network means it must be more complicated. More complicated devices are more difficult to harden properly to prevent successful attacks. The device must also have more computer power to handle the added functionality.

Router

The products that terminate their tunnel on routers or router-like devices (gates) are usually stand-alone proprietary hardware with embedded software. The device is connected in parallel to or in front of the firewall.

Advantages

The main advantages are speed and simplicity. The software is embedded in a special purpose device so relatively inexpensive hardware provides fast throughput with simple operations and maintenance. The manufacturer has hardened the device.

Concerns

The main disadvantage is that it is an additional device that is accessible from the Internet. For consulting firms that must support manyVPNs the additional devices can become a large maintenance cost.

Servers

The products that terminate their tunnels on servers are generally software-only solutions that use general-purpose servers (MS Windows or Unix) rather than proprietary hardware. The server is connected in parallel to, or in front of, the firewall.

Advantages

Software only solutions are generally less expensive. General-purpose hardware is typically less expensive to operate than proprietary hardware. For consulting firms that must support many VPNs, being able to install multiple VPNs on a single server can reduce the operating and maintenance costs.

Concerns

The main disadvantage is that there is an additional device that is accessible from the Internet, which the administrator must harden from attack

Security Services

Authentication

This service is a measure of how sure you can be of the identity of the person setting up the tunnel. The New Yorker published a cartoon a few years ago that depicted two dogs looking at a desktop computer. The caption read "On the Internet no one knows you're a dog". There are three basic types of authentication:

- Challenge/response (something you know)
- Token (something you have)
- Biometrics (something you are)

Challenge/Response

This is the most familiar authentication method. The user provides a name and password. Over the years, Remote Access Services (RAS) have developed a number of protocols based on this method. These include: PAP, CHAP, RADIUS, DIAMETER, and EAP.

Advantages

It's simple and inexpensive. No special hardware is required at the client or remote end.

Concerns

Its not very strong. Usernames are generally based on an algorithm such as last name/first initial. Once the algorithm is known, it is simple to guess someone's username. There are lots of password-cracking programs available on the Internet.

Token

This authentication method is based upon a 'token'that has been given to the user. The person attempting to set up the tunnel must demonstrate they have the token as well as knowing a name and password. A simple example of a token is a bankcard. Some geniusrealized that the average person carries so many credit and bank cards that they can't memorize all of the numbers on the cards. By requiring the number on the card to be entered before the connection is completed, it is possible to be confident that the person setting up the connection has the card in their possession.

Other common forms of tokens include public key certificates, smart cards and one-time password or code generators. Public key certificates are software that contain one or more encryption keys based on the X.509 standard. Smart cards also usually contain X.509 encryption keys, however, since they are stored on a memory chip, the user is more likely to know if the keys have been stolen. (The card is gone☺). One time passwords are usually based on the S/Key algorithm and code generators are special purpose computing devices (e.g. SecurID, SafeWord or CryptoCard) that calculate a large pseudo-random number.
Advantages
Tokens are usually considered strong authentication because you have to have something (the token) as well as know something (a password).
Concerns
Smart cards require that a special reader be attached to the client in order to read the card.
Certificates must be stored on the client computer limiting the versatility of the token.
One-time codes require an application at the protected site that can calculate the same number as the token that is handed out to the user.
A process must be developed and implemented to quickly replace lost tokens, as the person setting up the connection must have a token in their possession.

Biometrics

This authentication method is based uponthings that is intrinsically unique about an individual such as voice prints, fingerprints or retinal patterns.
Advantages
There is a high probability that a person identified by biometrics is who they claim to be although there are the scenes in *Terminator II* and *Judge Dread,* where a helpless victim is dragged to an information kiosk to provide identification. The Black Hat then disposes of the victim and proceeds to extract the information being sought.
Concerns
This method of authentication requiresspecialized equipment to be attached to the remote end of the connection.

Access Control

This service provides the capability of restricting in some manner what the user can access once they have been authenticated. The restriction can either be done on the basis of packet filtering, which restricts either certain types of packets, or packets with specific addresses, or it can be done on the basis of policies, which define parameters such as application, individual, group and/or subnet.

Administrators may need to identify the individual user, not just the address at which the tunnel starts. They may need to route the traffic coming through the tunnel in order to maintain network security. The ability to control exactly who has access to what resource on a network is a key characteristic of the level of security that the VPN provides.

Confidentiality

This service is a measure of how certain the user can be that unauthorized personnel cannot see the material. It is provided by a combination of strong encryption algorithms (128 bit RSA) and strong authentication. It is important to understand the level of confidentiality required. Obviously, applications that support private companies bidding on government contracts in countries that are known to be corrupt require a higher level of confidentiality than applications supporting general e-mail.

Protocols used to create a VPN

VPNs are commonly created at Layer 2, the link layer, Layer 3 the network layer, or layer 5, the session layer, of the OSI model[i]. Each of these protocols brings strengths and weaknesses to the VPN solution. The VPN Overview web page [3] provides a detailed comparison between VPNs built from different protocols. I have summarized and added to that information below.

Session Layer

Advantages

Session layer VPNs can provide more detailed control of the flow of data than VPNs based on lower layer protocols. They can work with a variety of security technologies for authentication and encryption. They establish a virtual circuit between the client and host on a session by session basis allowing monitoring and access control that is based on user authentication. They proxy traffic between the source and destination, so all data packets can be transferred through a single port on a firewall. Since session layerVPNs operate above the TCP/IP stack they dont require changes to the network drivers.

Concerns

The main disadvantage is that session layerVPNs proxy all traffic. This means that they are slower than lower layer VPNs. Their more sophisticated access control is more complicated to set-up manage and maintain than address based access control schemes.
The common layer 5 (Session) protocols are SSL, SOCKs and SSH.

- Secure Sockets Layer (SSL), which is an IETF/W3C standard designed to encrypt http traffic.
  Advantages
  The protocol is available in all modern browsers so special client software to initiate the tunnel to the proxy server is not needed.
  Concerns
  Since it uses the browser to establish the tunnel to the proxy server, it only works with web-based applications or applications that have been adapted to use a web-based interface (web based X11 or ICA).
- SOCKS is an IETF standard designed for authenticated firewall traversal. It is the protocol used in Aventail's VPN solution.
  Concerns
  It requires software on the remote computer to set-up the connection to the proxy server or special versions of software that are SOCKs-aware.
- Secure Shell (SSH2) is a draft IETF standard designed to provide secure remote shell commands. The Secure Shell program, which runs mainly on Unix computers, uses it.

Network Layer

The only Layer 3 (network) protocol used in VPNs is IPSec.

- Secure IP (IPSec) is the IETF standard for encrypting IP packets. It is included in Microsoft Windows 2000.
  Advantages
  IPSec is an extension of IP so there is the promise that it will be widely implemented in the VPN market. It supports a variety of encryption algorithms and checks the integrity of the transmitted data to ensure they have not been tampered with en route. Since it was designed to provide security between firewalls and routers it is an optimal solution for gate to gate VPNs.
  Concerns
  The main disadvantages are a result of the advantages noted above. First, because IPSec ensures the integrity of the data, it wont work from behind a Network Address Translation (NAT) firewall. It assumes that the address change means the integrity has been compromised. Because it was designed to work between gates, it does not have any routing capability built in and its authentication is based in part on the IP address so it identifies the device not the user. SinceIPSec is a relatively new protocol and allows a variety of encryption algorithms to be used, most vendor products do not inter-operate. Finally, it does not support protocols other than IP.

Link Layer

Advantages

Link layer VPNs were originally designed to extend Remote Access Services over the Internet. Their number one advantage is that they come as part of Microsoft Windows so it are free and already installed. Link layer protocols can tunnel additional protocols (IPX, NETBEUI, Appletalk, etc.) not just IP. They can provide flow control thus optimizing transmission by cutting down on dropped packets.

Concerns

Link Layer VPNs are targeted at the Microsoft client space. There are very few clients for other operating systems such as Unix,Linux or Macintosh.

The common Link Layer protocols are PPTP and L2TP.

- Point to Point Tunnelling Protocol (PPTP) is an industry standard, designed to tunnel PPP traffic within IP packets. It is included with Microsoft Windows NT4 and 98.
  Concerns
  There has been a lot of discussion on the web, news groups and mailing lists regarding the quality of the encryption and authentication provided by Microsofts VPN based on PPTP. The VPN FAQ web page [4] gives a balanced coverage of the issue. Some people do not consider the security to be adequate. Once a layer 2 tunnel is set up it is bi-directional as long as the tunnel stays up. No additional authentication is provided. Access control is based upon packet filtering.
- Layer 2 Tunnelling Protocol (L2TP) is an industry standard resulting from the marriage of PPTP and Ciscos Layer 2 Forwarding (L2F) protocol. It was designed to encapsulate IPX, AppleTalk and NetBEUI traffic within IP packets. It is included with Microsoft Windows 2000.
  Advantages
  It comes as part of Windows 2000. Microsoft has done a lot to eliminate the security concerns by supporting additional authentication facilities such

as RADIUS and tokens such as one-time codes and smart cards.

Cost

The cost of a VPN product is a function of the characteristics described above. Solutions which require something (software, token, specialized reader, etc) to be distributed to each user are more expensive than those that only require services already available in the client computer

For the same reason products that use dedicated hardware to terminate the tunnel are more expensive to purchase and maintain than products that simply add software to an existing firewall.

Effect of Windows 2000

Windows 2000 comes with a VPN client built in. RAS Services support PPTP, L2TP and L2TP over IPSec protocols and MS-CHAP, RADIUS, and smart card authentication methods. Most VPN manufacturers have revised, or are revising, their products to support this built-in client. In the future only special applications (e.g. government diplomatic services) will requirespecialized clients. These will typically be applications with very high authentication requirements.

TransCanada's solution

Strategy

TransCanada's VPN strategy comes in two parts. First, the main VPN will be based on the SSL protocol. Second, all non web-based traffic that passes over public networks will useIPSec.

Reasons for Choice

TransCanada wanted the VPN to run on stand-alone hardware, either a proprietary device or a Sun Unix server (company standard for servers). TransCanada did not want to add additional intense processing to a device whose original purpose was to control network access. Remember, the firewall is an intentional bottleneck on the network. This requirement eliminated all of the firewall vendors from the list.

TransCanada has chosen web-based delivery as its application display mechanism of choice. Over the next few years (target completion date - 2005), TransCanada will be converting all of its applications to a web-based user interface.
By providing a VPN based on HTTPS, all applications will be available toauthorized users no matter where or how they access the application. Information Technology won't have to be concerned about what client device is being used. In the spring of 1999, the only VPN based on HTTPS shipping was Sun's iPlanet, now known as iPlanet's Portal Server.

Internally, all traffic on TransCanada's networks will be IP packets. Therefore, an IPSec VPN will be used to extend TransCanada's networks through the Internet. However the requirement for IPSec compliance in a particular product does not by itself provide a means for selecting a particular VPN vendor solution. Remember the earlier statistic that more than 50+ vendors at the 1999 InterOp+Networld sported IPSec systems.

TransCanada identified a second requirement of relatively strong user authentication, which the company defined as "something you know, something you have and if you lose the thing you have, you know you've lost it." This requirement meant that either tokens or biometrics would be required to identify the person requesting a VPN tunnel be set up. Using biometrics and smart cards was deemed to be too costly, since special hardware had to be attached to each client. An X.509 certificate stored on the hard drive did not meet the requirement of knowing when it was lost, since the certificate could be copied without the owner's knowledge. This requirement did little to narrow the list as all of the IPSec VPNs at InterOp+Networld provided various mechanisms for performing user authentication. Fortunately, iPlanet's Portal server supports multiple tokens.

Both IPSec and HTTPS require Public Key Encryption to work. TransCanada had already implemented an Entrust Public Key Infrastructure (PKI), so another requirement was that the VPN had to work with the Entrust PKI. The certificates are used to identify the device that terminates the tunnel at TransCanada's network and to identify remote devices in situations where two networks are joined by a VPN, such as branch offices. In the spring of 1999 the only shipping VPN which integrated with the Entrust Certificate Authority (CA) was Timestep. IPlanet can use X.509v3 certificates. Entrust claims its CA will be X.509v3 compliant with its next version.

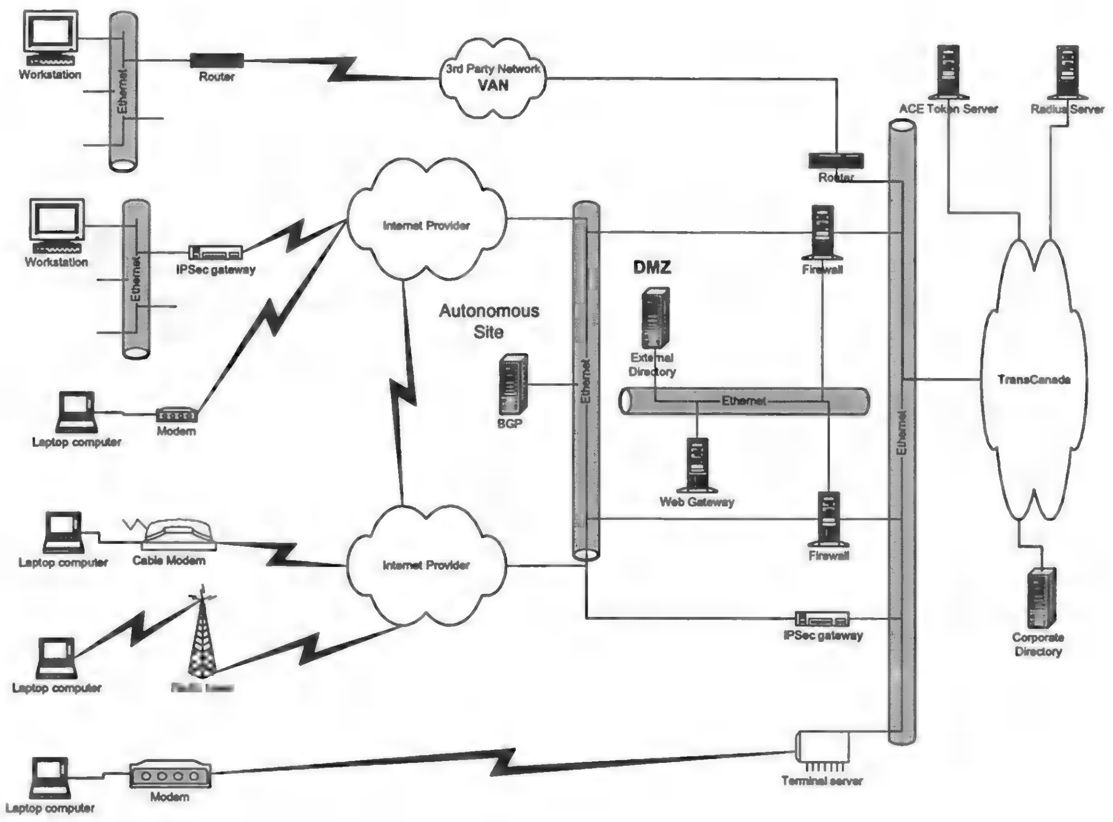


Figure 2 TransCanada External Access Architecture

Implementation

Figure 2 is a model of TransCanada's External Access Architecture. The Timestep VPN was put into production in April of 2000 after testing for six months. It uses the Corporate Drectory (a Netscape Enterprise Directory) to store Entrust Public Encryption Keys. Two Shiva Access Switches provide RAS services. The switches use RADIUS to authenticate users against the Corporate Directory. This service will not be going away any time soon as our 1-800 service is cheaper than a low speed ISP connection. Finally, the iPlanet web proxy service is currently in pilot mode. It will be placed into production by the end of the year.

Thus by the end of the year TransCanada will have a general purpose web-based VPN service that can be used with any browser, on any type of computer, and from any ISP connection. TransCanada will also have a special purpose IPSec VPN. This VPN will be used to selectively replace our existing dedicated lines in both our Intranet and our Extranet.

The replacement will be done on an as-needed basis meaning that if a change to an existing dedicated line is needed for business purposes (office moves, significant changes to bandwidth requirements, etc.) an analysis will be performed to see if the VPN can provide a more cost effective solution. The IPSec VPN will also allow TransCanada to extend its network to individuals' homes to support teleworkers at LAN speeds. Finally, there will be a low speed direct dial connection for those instances where Internet access is not available.

Appendix

Requests for Comment

Topic	RFC Number
Public Key Encryption	2407, 2408, 2409, 2459, 2510, 2511, 2527, 2528, 2559, 2585, 2587, 2560, 2692, 2693
IPSec	1828, 1829, 2104, 2085, 2401, 2410, 2411, 2402, 2412, 2451, 2403, 2404, 2405, 2406
L2TP	2661
SOCKS	1928, 1929, 1961
SSH2	In Draft
SSL	2246, 2712, 2716, 2817, 2818
S/Key	2243, 2289, 2444
CHAP	1994, 1472

Bibliography

1. Kosiur, Dave, *Building and Managing Virtual Private Networks*, Wiley Computer Publishing, 1998.
2. Ball, Larry L., *Cost-Efficient Network Management,* McGraw-Hill, 1992.
3. http://www.maximalsolutions.com/current/VPNs/VPNOverview.html, Virtual Private Networks - An Overview.
4. http://kubarb.phsx.ukans.edu/~tbird/vpn.html, Virtual Private Networks Frequently Asked Questions.
5. http://www.epm.ornl.gov/~dunigan/vpn.html, Tom Dunigan's Virtual Private Networks page.
6. http://www.mnmteam.informatic.uni-muenchen.de/projects/vlan/vpn.shtml, Virtual Private Networks.

[i] The International Standards Organisation (ISO) developed an Open Systems Interconnect (OSI) model of network communications in the 1980s. The model segmented the communications process into 7 functional layers for which standards of operation were written[2]

System Security Administration for NT

Harlan Carvey
Hcarvey@netsol.com

Network Solutions
http://www.networksolutions.com

Abstract

System security administration has, for the most part, been largely ignored as network administration has flourished. The result of this is that there are large installations of NT that need to be retrofitted with some form of security administration and management system. There are various third party tools available to assist in this endeavor, but they are somewhat general and not tailored to meet the needs of a particular user or organization. System administrators must therefore learn to mold their infrastructure to the tool, rather than the other way around. Often times, the tools may also be quite expensive, and difficult to learn and maintain.

1. Introduction

Security administration for a single NT system can be cumbersome, at best. There is no single interface for configuring and monitoring the security posture of an NT system. For example, the audit policy for a standalone NT system is set via the User Manager, while log-specific settings and all monitored activity is recorded in the EventLog. Further, each object (file, directory, share, Registry key) has it's own interface for enabling access control lists (ACLs) and audit settings. Rolling out a common audit standard across an NT enterprise, and then monitoring the EventLogs, can be a daunting task. However, there are many third-party commercial tools available to assist the system administrator. Two drawbacks of these tools are their vagueness and cost. Commercially available solutions are very general in nature, not providing the ability to focus in on the specific needs of the administrator. Some check for compliance with "best practices" but the definition of "best practices" is purely arbitrary. Other tools provide a wealth of information. In fact, they provide too much information, inundating the administrator with facts, requiring him to sift through this information and attempt to extract the real issues at hand. None of the tools take other aspects of the administrator's infrastructure into account, such as corporate security policies, the existence of a firewall, ACLs on routers, VLAN implementations on switches, etc. The cost of purchasing and installing the tools, coupled with the "cost" of deciphering the collected data, makes for a very low return on investment (ROI). Updates to the tools to look for new vulnerabilities may even be an added expense.

System administrators need a more viable solution. The appropriate solution for a security administration system must:

- Be simple, and easy to construct and maintain.
- Be centralized.
- Be scalable.
- Be configurable, to meet the specific needs of a user.
- Provide for verification of security policy compliance (compliance checking and verification).
- Notify the administrator of systems that are not in compliance.
- Where appropriate, modify system settings to bring systems into compliance with policies.
- Reduce overall cost for administering the system (i.e., no expensive per-seat license fees, etc).

1.1 Purpose

The purpose of this paper is to provide a framework, using Perl, for administering system security across an NT enterprise. Due to space limitations, all code examples cannot be presented in this paper. Additional scripts will be available via the author's web site at http://www.patriot.net/users/carvdawg/perl.html.

1.2. Background

The solution will be implemented in Perl, a scripting language that has been used on Unix platforms for administration purposes of years, and has been made available for the Win32 platform. ActiveState provides a fairly complete distribution of Perl for Win32 systems, as well as several avenues of support. There are also several additional modules that provide a quantum leap in functionality and usability, in the NT environment, to the ActiveState distribution. These modules provide a convenient wrapper around the Win32 API functions, making them more accessible, and easier to use and understand. This access to the Win32 API facilitates collection, and if necessary, modification, of security-relevant data for NT machines across the enterprise.

2. Solution

The solution for the centralized security administration of NT systems can be separated into three phases: collection, filtering/analysis, and modification. The three phases are kept distinctly separate in order to maintain simplicity, scalability and functionality. Separating the phases also makes it easier to construct a working set of tools, by allowing for testing and verification of one phase before moving on to the next. Additional functionality can be added to one phase without requiring any changes to the other phases.

The collection phase is used to extract raw configuration data from NT machines across the enterprise:

- Generate MD5 checksums of specific files to ensure file integrity.
- Registry settings (checking for security-specific settings, trojans, viruses, etc)
- File, directory, share and Registry key ACLs
- Services (running, stopped, and paused), the account each service runs under, and the executable called by the service
- Audit policies and EventLog entries
- User rights and user account information
- File (boot.ini) and directory ("Start Menu" for each user) contents

This data is saved in a central location, most likely on an administration workstation, in a suitable format. The data may be written to an Access database or an Excel spreadsheet, but the simplest method is to choose a standard delimiter, such as a colon or semi-colon, and write the data to flat text files. The data is simple to parse and review, and easily compressed and encrypted for archival. The examples in this paper will send all data to STDOUT, allowing it to also be redirected to a flat file.

The data can be easily analyzed using filters written in Perl, making use of its inherent ability to parse data and files. These filters access and process the data based on predefined conditions, making full use of the pattern matching abilities of regular expressions. Filters may be designed to check for:

- Known security issues, such as ACLs (on files, directories, and Registry keys) or specific Registry values.
- More complex issues, such as the ACLs on a file as it's related to the function of the particular system (PDC, BDC, or workstation) and a Registry key entry.
- General trends across the enterprise, such as the level of Service Packs and HotFixes installed.
- Compliance with corporate security policies, to include the audit policy for each system, or the existence of modems on workstations vice RAS servers.

The processed information can then be presented in whichever format meets the needs of the user, to include:

- Email
- Pager
- HTML
- XML
- Word document
- Excel spreadsheet
- Access database
- EventLog entries

The system administrator can also use Perl to act upon the information derived from filtering and analysis, making modifications as necessary. Services can be stopped or started, Registry values added or deleted, auditing can enabled, and EventLogs can be collected and filtered, all from a single workstation.

Due to it's simple, modular nature, this system can be easily updated as new information (new vulnerabilities, updated security policies) is incorporated.

This system can also be used to establish a baseline, which managers and administrators can use to look for system changes, or develop and implement security policies and standards.

Once the data is filtered and analyzed, system administrators can begin the modification phase. Using the same Perl modules that were used to collect data from systems, the system administrator can bring the NT systems into compliance with security standards, by updating settings (i.e., Registry entries, file/directory/share/Registry permissions, audit policy, etc). As this is done from a centralized location, only one account, with a strong password, is required to perform this entire range of security administration.

Once all settings have been implemented and tested, the collection and filtering phases can again be used to verify compliance with security standards, by executing those phases as part of a regular security scan. Using the baseline information, the system administrator need only to look for anomalies and unusual settings, indicating that a lack of compliance with security policies and standards.

As an example, assume the system administrator wishes to not only check for the existence of trojan programs on her NT servers and workstations, but she also wants to ensure that these programs (or any other programs, for that matter) can not be installed on the systems. She uses Perl scripts collect relevant data (Registry key entries and permissions, file and directory ACLs, etc) from across the enterprise, to include offices connected to the master domain by WAN and VPN links. Once she's collected data from all systems, she filters the Registry entries based on published signatures of known trojan programs, and then reviews the entries for suspicious or unauthorized entries. Not finding anything significant, she then sets about modifying Registry and directory ACLs on all systems to prevent the installation of trojan programs in the future. She follows this up with periodic scans of the Registry keys and Security EventLog to ensure compliance, and incorporates new data, as it becomes available. The entire process is simple to maintain, and automated to reduce errors and increase efficiency.

3. Implementation

The examples presented in this paper are not meant to be all-inclusive. Rather, they are presented as a framework or set of tools from which system administrators can build solutions that meet their specific needs. Also, space requirements limit the number and functionality of the scripts that can be presented. Additional scripts will be available via the web, at http://www.patriot.net/users/carvdawg/perl.html.

The following examples lay the groundwork from which specific solutions and systems can be built.

3.1 Checksums

The integrity of critical system files is a serious security concern, as there are publicly available web sites that demonstrate the construction of a rootkit for NT (http://www.rootkit.com). Rootkits have been a thorn in the sides of Unix administrators, as they are used to patch the kernel and trojanize commands in order to hide malicious activities. DIGEST.PL (Script 1) demonstrates a method for generating MD5 hashes of critical files. These hashes can be saved to a protected file or database and be used to verify the integrity of critical system files on a regular basis.

3.2 Registry Settings

The NT Registry can be as much a treasure trove as it is a technological minefield. There are several Registry values that have a direct or indirect impact on the security posture of the system.

3.2.1 Individual Registry Values

REGKEYS.PL (Script 2) shows how Registry values can be collected from remote NT systems. The values can then be verified, and a message printed to STDOUT for each incorrectly set Registry value. This information can be logged to a database or file, or the value simply modified as needed. For example, the system administrator can check NT systems for compliance by checking the Service Pack level on each system.

3.2.2 Trojan Keys

TROJANKEYS.PL (Script 3) demonstrates collecting the contents of Registry keys. The values can be scanned for known trojan signatures using the grep() function. Trojan signatures are available from anti-virus sites such as F-Secure (http://www.f-secure.com). For example, using grep(), the system administrator can quickly scan incoming data using the following signatures for network backdoor, remote administration trojan programs (signature in parens):

- ATAKA (SNDVOL.EXE)
- NetBus (Patch)
- DeepThroat (SystemDLL32)
- NetSphere (NSSX)
- GateCrasher (Command)
- Portal of Doom (ljsgz.exe)
- Girlfriend (Windll.exe)
- HackATack (Expl32.exe)
- Phase Zero (MsgServ|msgsvr32.exe)
- MyPics Variant (agent5|zip01.exe)
- NewApt (tpawen|panth)
- SubSeven (mtmtask.dl)

By simply changing the Registry key examined, system administrators can verify which HotFixes are installed on each NT system.

3.3 ACLs

Access control lists (ACLs) define which users and groups have what level of access to specific resources. PERMS.PL (Script 4) demonstrates a method for collecting the ACLs from files, directories, Registry keys, and shares, and presenting the ACLs in an easy to understand format. Credit goes to Dave Roth for both his Win32::Perms module, as well as the code for the getperms() method.

3.4 Services

SERVICES.PL (Script 5) demonstrates a method for collecting information from services available on NT systems, such as the service's status, the service executable, and the account that the service runs under. Other methods in the Win32::Lanman module will allow the system administrator to start or stop services. Dave Roth's Win32::Daemon module provides a nice interface for creating services from Perl scripts.

3.5 Auditing and Logging

NT systems have the capability of performing auditing and logging functions. Administrators can use the information available in the EventLog to see if there are problems with applications, look for clean or dirty system shutdowns, or even as a rudimentary intrusion detection system.

3.5.1 Audit Policies

The AUDITPOL.PL (Script 6) script shows how the audit policy can be determined. Auditing can be enabled if the current state shows it to be disabled, and the particular events to be audited can be set.

3.5.2 EventLog Entries

The DUMPEVENTS.PL (Script 7) script demonstrates a means of collecting EventLog entries from multiple machines within the domain. The script collects the entries and sends them to STDOUT. However, the entries can be deposited in an Excel spreadsheet or Access database, making the entries more manageable. Specific events can be filtered during the collection process, such as Dr. Watson messages, failed logon attempts, etc. If ACLs are set to prevent users from writing to specific directories and Registry keys, attempts to do so (as when the user attempts to install software, or when a trojan attempts to install itself) appear as EventID 650 in the Security EventLog.

3.6 User Privileges and User Account Information

The administrator may need to determine the privileges that users have available in order to diagnosis access issues. The Win32::Lanman module provides methods for displaying privileges assigned to particular users or groups, or assigning privileges as necessary.

USERS.PL (Script 8) demonstrates how an administrator might collect information about user accounts from the local NT system. Other Win32::Lanman methods allow the administrator to enumerate global and local groups, or add users and groups.

3.7 Directory and File Contents

STARTUPCHK.PL (Script 9) shows how an administrator can examine the contents of the Start Up directories for all profiles on an NT system. This script will also show the executable files pointed to by shortcuts.

Administrators can also use Perl to get the contents of particular files, such as boot.ini or lmhosts, in order to aid in troubleshooting problems.

4. Conclusions

The system and example scripts presented in this paper are quite simple, yet scalable and easy to maintain. All requirements for a security administration system have been met, to include reducing overall cost. At the same time, system administrators using this model have an understanding of the necessity for maintaining the security of their NT systems. They also have a complete security administration toolkit consisting of resources for NT-specific security information, a powerful scripting language (Perl), and a wide variety of tools and techniques to make their jobs easier and more efficient.

Using other modules, system administrators can perform a wide variety of other centralized security administration tasks. For example, there are modules available that allow the system administrator to interface with the Performance Monitor on remote NT systems to check running processes. System administrators can also use Win32::OLE to interface locally with IIS 4.0's metabase and make configuration changes, or use LWP::UserAgent to test their IIS 4.0 web servers for susceptibility to the latest exploit (note:

Rain Forest Puppy's whisker.pl is uses Socket.pm to provide the functionality of LWP::UserAgent).

5. Appendix A: Resources

5.1 Perl on NT

All scripts associated with this paper were written and tested using ActiveState's ActivePerl build 522 on Windows NT 4.0. At the time of this writing, ActivePerl build 522 was available from ftp://ftp.activestate.com/ActivePerl/Windows/5.005/Intel/.

5.2 Extensions

The following extensions are available as part of the default distribution of ActiveState's ActivePerl:

- Win32::TieRegistry
- Win32::Shortcut

The Digest::MD5 module is not installed as part of the standard distribution, but is available via PPM ("ppm install Digest-MD5").

The following extensions are available from Dave Roth's web site, and installable via PPM:

- Win32::AdminMisc
- Win32::Perms

The Win32::Lanman module is available in a zipped archive from Dave Roth's FTP site (ftp://ftp.roth.net/ntperl/Others/Lanman/lanman_1_05.zip) or it can be installed via PPM from http://jenda.mccann.cz/perl/.

5.3 Books and Papers

Schwartz, R. L., Olson, E., and Christiansen, T., "Learning Perl on Win32 Systems", O'Reilly & Associates, 1997

Roth, D., "Win32 Programming: The Standard Extensions", MacMillian Technical Publishing, 1998

Roth, D., "A Network Machine Management System", USENIX's 2nd Large Installation System Administration of Windows NT (LISA-NT) *Conference*, July, 1999

Jumes, J. G., Cooper, N. F., Chamoun, P., and Feinman, T. M., "Windows NT 4.0 Security, Audit, and Control", Microsoft Press, 1999

5.4 Web Sites

The author's web site lists several scripts which could not be listed in this paper due to space requirements (http://www.patriot.net/users/carvdawg/perl.html).

Dave Roth's site is the home of several extremely useful Perl modules, as well as his paper from the Usenix LISA-NT '99 Conference (http://www.roth.net)

Joe Casadonte's web site lists many modules available for NT (http://www.netaxs.com/~joc/perlwin32.html)

6. Appendix B: Scripts

6.1 Digest.pl (Script 1)

```
#! c:\perl\bin\perl.exe
# digest.pl
# Generate MD5 checksums on files to verify file
# integrity
# Use with system files, web pages, etc.  Can also use
# files on mapped drives, but must use complete path.
use strict;
use Digest::MD5;

my $server = shift || Win32::NodeName;
my @files = ("io.sys","ntldr","boot.ini","ntdetect.com",
             "winnt\\system32\\fpnwclnt.dll");
my $path;
$server =~ y/a-z/A-Z/;
($server eq Win32::NodeName) ? ($path = "c:\\") :
 ($path = "\\\\$server\\c\$\\");

foreach (@files) {
 my $file = $path.$_;
 if (-e $file) {
  my $hash = hash($file);
  print "$server\t$file => $hash\n";
 }
 else {
  print "$file could not be found.\n";
 }
}

# sub() to generate the actual hash
sub hash {
 my ($file) = @_;
 open (FILE, $file) or die "Can't open $file: $!\n";
 binmode(FILE);
```

```
  my $digest = Digest::MD5->new->
      addfile(*FILE)->b64digest;
  return $digest;
}
```

6.2 Regkeys.pl (Script 2)

```
#! c:\perl\bin\perl.exe
# RegKeys.pl
# Collect Registry key values for analysis
use strict;
use Win32::TieRegistry(Delimiter=>"/");

my $server = shift || Win32::NodeName;
my $remote;

if ($remote = $Registry->{"//$server/LMachine"}) {
  \&getRegValues($server);
}
else {
   print "Could not connect to $server Registry.\n";
}

sub getRegValues {
  my($value,$data);
  my %regkeys = ();
# Keys are kept in a datafile, in format: Value;Path
# Ex: EnableDCOM;SOFTWARE/Microsoft/Ole
  my $datafile = "RegKeyValues";
  if (-e $datafile) {
    open(FL,$datafile) || die "Could not open ".
       " $datafile: $!\n";
    while(<FL>) {
      chomp;
# Skip comments and blank lines
      next if ($_ =~ m/^#/);
      next if ($_ =~ m/^\s+$/);
       my($key,$path) = split(/;/,$_);
       $regkeys{$key} = $path;
     }
      close(FL);
      foreach my $key (keys %regkeys) {
        $value = $remote->{$regkeys{$key}};
        $data = $value->{$key};
        if (defined $data) {
          $data = hex($data) if ($data =~ m/^0x/);
          print "$key = $data\n";
        }
         else {
            print "$key = NotFound\n";
        }
      }
    }
}
```

6.3 TrojanKeys.pl (Script 3)

```
#! c:\perl\bin\perl.exe
# trojankeys.pl
# Check Registry keys where trojans
# like to hide for suspicious entries
use strict;
use Win32::TieRegistry(Delimiter=>"/");

my ($remote);
my $server = shift || Win32::NodeName;

my %hives = ("LMachine" => "HKLM",
             "CUser" => "HKCU");

foreach my $hive (keys %hives) {
  print "Checking $hives{$hive} hive...\n";
  ($remote = $Registry->{"//$server/$hive"}) ?
    (\&getTrojanKeys($server)) :
    (print "Could not connect to ".
      " $hives{$hive} hive.\n");
   print "\n";
}

sub getTrojanKeys {
  my($path) = 'SOFTWARE/Microsoft/'.
    'Windows/CurrentVersion';
  my(@keys) = ('Run','RunOnce','RunOnceEx',
                'RunServices');
  foreach my $k (@keys) {
    my $key = $remote->{"$path/$k"};
    if (defined $key) {
      my @vals = $key->ValueNames;
      if($#vals != -1) {
        foreach my $val (@vals) {
          my $data = $key->{$val};
          $data = "NotFound" unless($data);
          print "$k:$val:$data\n";
        }
      }
    }
  }
}
```

6.4 Perms.pl (Script 4)

```
#! c:\perl\bin\perl.exe
# Perms.pl
# usage: perl perms.pl [obj]
# ex: perl perms.pl c:\winnt
use strict;
use Win32::Perms;

my $obj = shift ||
 die "You must enter an object: File, Dir,".
     " or Reg key.\n";

\&getperms($obj);

# Credit goes to Dave Roth for providing the
# following code to translate the DACL mask
# into something readable; ie, Explorer-like
# listing
sub getperms {
  my($obj) = @_;
  my($Acct,@List,$Path,$iTotal,@String);
  my($Perm) = new Win32::Perms($obj);

  my (%PERM) = (R  => 0,
                W  => 1,
                X  => 2,
                D  => 3,
                P  => 4,
                O  => 5,
                A  => 6);

   my(%MAP) = ('FILE_READ_DATA'    => 'R',
               'GENERIC_READ'      => 'R',
               'KEY_READ'          => 'R',
               'KEY_QUERY_VALUE'   => 'R',
               'FILE_WRITE_DATA'   => 'W',
               'KEY_WRITE'         => 'W',
               'GENERIC_WRITE'     => 'W',
               'KEY_SET_VALUE'     => 'W',
               'DELETE'            => 'D',
               'FILE_DELETE_CHILD' => 'D',
              'FILE_EXECUTE'       => 'X',
               'FILE_TRAVERSE'     => 'X',
               'GENERIC_EXECUTE'   => 'X',
               'CHANGE_PERMISSION' => 'P',
               'TAKE_OWNERSHIP'    => 'O',
              'FILE_ALL_ACCESS'    => 'A',
               'GENERIC_ALL'       => 'A',
               'STANDARD_RIGHTS_ALL'
                 =>'A');

  die "Can not obtain permissions for '$Path'\n"
       if(!$Perm);

  $Perm->Dump(\@List);

  foreach $Acct (@List) {
    my($Perm);
    my(@String) = split(//, "-"
         x scalar(keys(%PERM)));
    my($Mask,@M,@F);
    my($DaclType);
    next if($Acct->{Entry} ne "DACL");
    $iTotal++;

    DecodeMask($Acct,\@M,\@F);
    foreach $Mask (@M) {
      $Perm |= 2**$PERM{$MAP{$Mask}};
    }

    foreach $Mask (keys(%PERM)) {
      $String[$PERM{$Mask}] = $Mask
        if ($Perm & 2**$PERM{$Mask}) ;
    }

    $DaclType = $Acct->{ObjectName};
    if( 2 == $Acct->{ObjectType} ) {
# We have either a file or directory. Therefore we
#  need to figure out if this
# DACL represents an object (file) or a
# container (dir)...
      ($Acct->{Flag} & DIR) ?
        ($DaclType = "Directory") :
        ($DaclType = "File");
    }
    my $permstr = join("",@String);
    print "$Acct->{Account}:$DaclType:".
      "$permstr\n";
  }
  print "Everyone has full permissions.\n" if(!$iTotal);
}
```

6.5 Services.pl (Script 5)

```
#! c:\perl\bin\perl.exe
# Services.pl
# Retrieve info on services on $server
use strict;
use Win32::Lanman;

my $server = shift || Win32::NodeName;

\&Services($server);

sub Services {
  my($server) = @_;
# Array to translate the current state of the service into
# something readable
  my(@state) = ("","Stopped","Start_Pending",
         "Stop_Pending",  "Running",
         "Continue_Pending",
         "Pause_Pending","Paused");

# Array to translate the startup options for the service
# into something readable
  my(@startup) = ("","","Automatic","Manual",
    "Disabled");
  my($err,@services,$service,%info);
  if (Win32::Lanman::EnumServicesStatus(
    "\\\\$server","",&SERVICE_WIN32,
   &SERVICE_STATE_ALL,\@services)) {
   foreach $service (@services) {
     if (Win32::Lanman::QueryServiceConfig(
       "\\\\$server","",${$service}{name},
       \%info)) {

# Print out in an easy to read format
        print "${$service}{display}\n";
        print "\tName\t${$service}{name}\n";
        print "\tState\t".$state[${$service}{state}]."\n";
        print "\tAccount\t$info{account}\n";
        print "\tFile\t$info{filename}\n";
        print "\tStart\t".$startup[$info{start}]."\n";
        print "\n";
      }
      else {
        $err = Win32::FormatMessage
          Win32::Lanman::GetLastError();
        $err = Win32::Lanman::GetLastError()
          if ($err eq "");
        print "$server: QueryServiceConfig Error: $err";
      }
    }
  }
  else {
   $err = Win32::FormatMessage
     Win32::Lanman::GetLastError();
   $err = Win32::Lanman::GetLastError()
     if ($err eq "");
   print "$server: Services Error: $err";
  }
}
```

6.6 AuditPol.pl (Script 6)

```
#! c:\perl\bin\perl.exe
# auditpol.pl
# Determine the audit policy of an NT system
use strict(vars);
use Win32::Lanman;

my $server = shift || Win32::NodeName;

\&AuditPol($server);
sub AuditPol {
  my($server) = @_;
  my(%info);
  my(%hash) = (
  "AuditCategoryLogon" => "Logon and Logoff",
  "AuditCategoryObjectAccess" => "File and
     Object Access",
  "AuditCategoryPrivilegeUse" => "Use of
     User Rights",
  "AuditCategoryAccountManagement" =>
     "User/Group Mgmt",
  "AuditCategoryPolicyChange" => "Security Policy
     Changes",
  "AuditCategorySystem" => "Restart and
     Shutdown System",
  "AuditCategoryDetailedTracking" =>
     "Process Tracking");

  my(%settings) = (0 => "None",
                   1 => "Success",
                   2 => "Failure",
                   3 => "Success and Failure");

if (Win32::Lanman::LsaQuery-
  AuditEventsPolicy("\\\\$server",\%info)) {
   my $audit = $info{auditingmode};
   if (!$audit) {
    print "$server: Auditing is NOT enabled!\n";
   }
   else {
    print "\n";
    printf "%-35s %-20s\n","Audit Events","Settings";
    printf "%-35s %-20s\n","------------","--------";
    my $options = $info{eventauditingoptions};
    foreach my $key (keys %hash) {
      printf "%-35s %-20s\n",
        $hash{$key},$settings{$$options[&$key]};
    }
   }
 }
 else {
  my $err = Win32::FormatMessage
    Win32::Lanman::GetLastError();
   print "$server: Audit Error:  $err\n";
 }
}
```

6.7 DumpEvents.pl (Script 7)

```
#! c:\perl\bin\perl.exe
# Dumpevents.pl
use strict;
use Win32::Lanman;
use Win32::Perms;
my $server = shift || Win32::NodeName;
Win32::Perms::LookupDC(0);

\&GetEvents($server,"Security");
```

```
sub GetEvents {
 my($server,$evtlog) = @_;
 my(@events,$event,$desc);
 my %types = (1 => "(Error)",
              4 => "(Information)",
              8 => "(Success Audit)",
              16 => "(Failure Audit)");

 my %category = (0 => "(None)",
                 1 => "(System Event)",
                 2 => "(Logon/Logoff)",
                 3 => "(Object Access)",
                 4 => "(Privilege Use)");

if(Win32::Lanman::ReadEventLog("\\\\$server",
 $evtlog, 0xffffffff, 0, \@events)) {
 foreach $event (@events) {
  print"Computer: ".${$event}{computername}."\n";
  print "Category:".${$event}{eventcategory}."
      ".$category{${$event}{eventcategory}}."\n";
  my $id = (${$event}{eventid} & 0xffff);
# Use the EventID to filter on specific types of events
  print "Event ID:   ".$id."\n";
  print "EventType: ".${$event}{eventtype}."
      ".$types{${$event}{eventtype}}."\n";
  print "Source: ".${$event}{source}."\n";
  print "SourceName:  ".${$event}{sourcename}."\n";
  print"Generated:   ".localtime(${$event}
   {timegenerated})."\n";
  print "Written: ".localtime(${$event}
   {timewritten})."\n";
  print "Flags: ".${$event}{reservedflags}."\n";
  print "User: ".Win32::Perms::
    ResolveAccount(${$event}{usersid})."\n";
  print "Description: ";
  if (Win32::Lanman::GetEvent
   Description("\\\\$server", $event)) {
    $desc = ${$event}{eventdescription};
    print $desc."\n";
  }
  else {
    my $strings = ${$event}{strings};
    print "\n";
    foreach (@$strings) {
     print "\t+".$_."\n";
    }
  }
# print "Data:".unpack("H".2*length(${$event}
# {data}), ${$event}{data})."\n"
#    if (${$event}{data} ne "");
   print "\n\n";
 }
}
 else {
```

```
   my $err = Win32::FormatMessage
      Win32::Lanman::GetLastError();
   $err = Win32::Lanman::GetLastError()
      if ($err eq "");
    print "$server:  ReadEventLog error: $err.\n";
  }
}
```

6.8 Users.pl (Script 8)

```
#! c:\perl\bin\perl.exe
# Users.pl
# Get user info from local system
use strict;
use Win32::Lanman;

my $server = shift || Win32::NodeName;
print "$server users...\n";

\&GetUserInfo($server);

sub GetUserInfo {
 my($server) = @_;
my(@users,$user,$pwage);

 if (Win32::Lanman::NetUserEnum
   ("\\\\$server",0,\@users)) {
  foreach my $user (@users) {
   $pwage = (split(/\./,${$user}{'password_age'}))
     /(3600*24);
   print "${$user}{'name'}\n";
   print "\tComment     => ${$user}{'comment'}\n";
   print "\tUID         => ${$user}{'user_id'}\n";
   print "\tPasswd Age  => $pwage\n";
#  print "\tLogon Svr   =>
#     ${$user}{'logon_server'}\n";
   print "\tLast Logon  =>
     ".mlocaltime(${$user}{'last_logon'})."\n";
   print "\tLast Logoff =>
     ".mlocaltime(${$user}{'last_logoff'})."\n";
   print "\tAccount does not expire.\n"
     if (${$user}{'acct_expires'} == -1);
   print "\tACCOUNT DISABLED.\n"
     if (${$user}{'flags'} &
       UF_ACCOUNTDISABLE);
   print "\tUser cannot change password.\n"
     if (${$user}{'flags'} &
     UF_PASSWD_CANT_CHANGE);
   print "\tAccount is locked out.\n"
     if (${$user}{'flags'} & UF_LOCKOUT);
   print "\tPassword does not expire.\n"
    if (${$user}{'flags'} &
    UF_DONT_EXPIRE_PASSWD);
   print "\tPassword not required.\n"
```

```
      if (${$user}{'flags'} &
      UF_PASSWD_NOTREQD);
    print "\n\n";
   }
  }
  else {
   my $err = Win32::FormatMessage
     Win32::Lanman::GetLastError();
    $err = Win32::Lanman::GetLastError()
      if ($err eq "");
    print "NetUserEnum Error:  $err\n";
  }
}

sub mlocaltime {
  ($_[0] == 0) ? (return "Never") :
    (return localtime($_[0]));
}
```

6.9 StartUpChk.pl (Script 9)

```
#! c:\perl\bin\perl.exe
# StartUpChk.pl
# Checks contents of StartUp folder
# for each profile
use strict;
use Win32::Shortcut;

my $server = shift || Win32::NodeName;
my $me = Win32::NodeName;

my $startdir;
($server eq $me) ? ($startdir = "c:\\") :
  ($startdir = "\\\\$server\\c\$\\");

my $start = $startdir."winnt\\profiles\\";
my $startup = "\\start menu\\programs\\startup";
my ($dir,$err);

if (-e $start && -d $start) {
  opendir(ST,"$start");
  foreach $dir (sort readdir(ST)) {
    next if ($dir eq "." || $dir eq "..");
    my $dir2 = $start.$dir;
    if (-e $dir2 && -d $dir2) {
      my $newdir = "$start".$dir."$startup";
      if (-e $newdir && -d $newdir) {
        opendir(SUP,$newdir);
            my @files = readdir(SUP);
            closedir(SUP);
            if (@files) {
              foreach my $file (@files) {
                next if ($file eq "." || $file eq "..");
                if ($file =~ m/lnk$/) {
                  my $shortcut = Win32::Shortcut
                    ->new($newdir."\\".$file);
                ($shortcut) ? (print
                  "$dir:$file:$shortcut->{Path}\n") :
                  (print "Error with
                  Shortcut: ".Win32::FormatMessage
                  Win32::GetLastError."\n");
                }
                else {
                  print "$dir:$file\n";
                }
              }
            }
        }
        else {
          print "$newdir does not exist or is not".
            " a directory.\n";
        }
      }
      else {
        print "$dir2 does not exist or is not a".
          " directory.\n";
      }
    }
  closedir(ST);
}
else {
  print "$start does not exist or is not a directory.\n";
}
```

On Designing a Database for Integrated User Management: Pitfalls and Possibilities

Amy LaMeyer

Shankaranarayanan Ganesan

Jesper M. Johansson

Boston University

Abstract

Decisions on implementing IT systems have often been departmental or isolated in nature. As a result many organizations now are faced with the challenge of integrating different networks and computers (in different departments or possibly even within each), each managed by a different operating system, and each running different types of applications. In the last decade all organizations have understood the importance of integrating the different systems and applications. While practitioners and researchers have proposed and implemented several methods for integrating systems, applications, and data, one area has been and continues to be difficult to integrate - user accounts.

In this paper we present an approach to integrate user account information from several systems. This approach is interesting for several reasons. First it is not specific to managing users on a particular system and is general enough to be used in an integrated environment that includes several systems of different types. Second, it is easily scalable to include applications and networks that have users and need user management. Third, it is simple and easy to understand and implement. The approach results in a database of user accounts that can be queried for specific users, groups, and so on. The design can be implemented in any relational database, or for example in the Windows 2000 Active Directory to take advantage of the fact that it already tracks user accounts and related information.

1. Introduction

Decisions on implementing IT systems have often been departmental or isolated in nature. As a result many organizations now are faced with the challenge of integrating different networks and computers (in different departments or possibly even within each), each managed by a different operating system, and each running different types of applications. In the last decade all organizations have understood the importance of integrating the different systems and applications. While practitioners and researchers have proposed and implemented several methods for integrating systems, applications, and data, one area has been and continues to be difficult to integrate - user accounts.

There may be several reasons why user account integration has not received a great deal of attention. Every enterprise system supports the management of users on that system. As a matter of fact, it is a basic security requirement (C1) that the OS be capable of identifying users (Department of Defense 1985). To further complicate matters, many applications, such as databases, define their own user accounts. Many systems have sophisticated user account databases that can store information not just about the user accounts, but comments about the users as well. Windows NT 4.0, for example, supports this, and the schema used to store user accounts in Windows 2000 can be extended to support a much richer set of information about each user. On the other hand, it is not at all uncommon to find systems that do not even support storing a user's real name, not to mention critical identifying information such as employee ID. This invariably results in an extensive duplication of user information, and the inability to answer specific question such as what access privileges a specific user of a system has on other systems in the organization, or even fundamental questions such as which employees have access rights on a specific system. Employees may also have different user account names on different systems. Organizations have often chosen not to deal with this problem, as they understand it to be vicious cycle. If the organization could track all of its IT users across all systems user management would be considerably simpler. To resolve this issue many organizations see the need to manage users in an integrated fashion, which is difficult, if not impossible, given the obscure status of current user information. The problem has therefore not been addressed and no feasible, practical solution suggested.

In this paper we present an approach to integrate user account information from several systems. This approach is interesting for several reasons. First, it is not specific to managing users on a particular system and is general enough to be used in an integrated environment that includes several systems of different types. Second, it is easily scalable to include applications and networks that have users and need user management. Third, it is simple and easy to understand and implement. The approach results in a database of user accounts that can be queried for specific users, groups, and so on. The design can be implemented in any relational database, or, for example, in the Windows 2000 Active Directory to take advantage of the fact that it already tracks user accounts and related information.

1.1 User Accounts

Each system (computers or networks based on NT, UNIX, VMS, etc.) supports ways of managing users on that system. Microsoft's Active Directory is designed to support the management of users enterprise-wide in a Windows-2000 server-based network, for example. Specifically, Active Directory addresses the management of users in terms of the privileges and resources available to or accessible by each. Other software firms have created and made available software to comprehensively manage users in a specific type/network of computers such as UNIX (e.g. COSadmin by Open Systems Management). It is important to note that all of the above address user management in a single, specific system or network. There are approaches that present exceptions, such as Novell's Netware Directory Services (NDS), which does work both on Netware and Windows NT/2000. However, it is still limited to managing user accounts on those systems for which it was designed. Our approach does not substitute these user management features but supplements them and helps pull them together by providing a design for a central location to track user accounts.

1.2 Identifying User Accounts

To completely understand the issues involved, let us briefly visit these. Individual computer systems typically have users identified either directly by a username (as in UNIX based implementations) or by a unique user identifier (the SID on NT based systems). Individual systems may be part of one or more computer network(s) and these have users who need to be managed. Further, large shared applications such as an application database, email, knowledge management subsystems etc. may use the user account database of the host OS, or support their own accounts. Some databases, such as Microsoft's SQL Server, support both approaches. Very often, however, the access to such applications is independent of the access to the computer systems they run on. Most of the above systems manage privileges assigned to users using groups or roles. We now have different systems that users have access to, different groups within each system that define and manage user privileges, different applications within each system that are accessed by users, and networks/sub-networks that connect these systems together. The proposed design for managing users in an integrated fashion addresses a majority of the above issues and can be extended to deal with the rest as well. We do not deal with privileges and access to specific resources in this paper, as individual systems provide good support for managing privileges and resources within each. As a first step, we address the other user management issues that are not adequately supported.

The remainder of the paper is organized as follows. The specific problem that motivated this paper is described in detail in Section 2. For convenience, and to protect the identity of the organization, we henceforth refer to the organization as Information Systems Inc. (ISI. Also described in Section 2 are the existing systems and corresponding user accounts, and the need for integrating the user management as well as specific issues that need to be addressed in this process. Section 3 presents a design of a database that incorporates existing accounts and integrates the management of users in the organization. A discussion of how to utilize some of the features of Windows 2000 to make the user identification task easier, are presented in Section 4 and the issues that need to be addressed to resolve problems in importing the existing data into this schema are described here. Concluding remarks, directions for improvement, as well as directions for further work are presented in Section 5.

2. The Current State of the Systems and User Accounts at ISI

ISI Inc. is an organization that provides a variety of high quality health care solutions. These are directed towards individual customers as well as for use in hospitals and home health care services. The categories of products include pharmaceutical, nutritional, diagnostic, and hospital supplies. ISI Inc. owned and operated a subsidiary organization that manufactured, sold, and more importantly provided customer service and support for some of the products manufactured at ISI that required customer service and post-sales support. ISI Inc., like any other organization today, uses IT and IT systems to store and manage their extensive data resources as well as maintain the applications that capture and make use of this data. Typical applications include the standard corporate applications for sales force planning, sales and demand forecasting, financial applications, human resource management, customer service and support, as well as applications for monitoring and planning their manufacturing and production processes. The human resource management applications maintain data on over 500 employees and more than 60 percent of their employees (> 300 employees) required and were given access to one or more computer systems and applications that were part of the IT infrastructure at ISI Inc.

The IT infrastructure at ISI Inc. included a variety of computer systems and applications. Some of these systems were integrated together in a networked environment while others were standalone. The corporate applications (CAS) resided on a Microsoft Windows NT-based client server network that included a set of desktop computers running Windows NT. The customer data and customer service applications are maintained on a system of Oracle databases that we shall refer to as Customer Service System (CSS). The production data is also captured on a different system of Oracle databases and we will refer to this as the Production Data System (PDS) for convenience. The manufacturing applications and data are managed by a manufacturing application system (MAS) installed on an IBM AS/400. The human resource (HR) data is maintained in a spreadsheet application (MS-Excel) and is part of the data that resides in the NT network for corporate applications. Each system is part of a specific functional unit within the organization. The manager of a functional unit is responsible for authorizing and managing the users in the system that is part of that functional unit.

The user-management was performed at two levels: the system level and the application level. The hardware/platform specifically, the operating system for each hardware/platform controlled the access for each user at the system level. This defined the authentication of the individual users, the files and directories within each system that the user had access to, and the read, write, and execute privileges each user held for the different applications that resided on this system. This is accomplished by creating groups/roles, assigning a set of permissions and privileges for each, and enrolling each user as being part of the appropriate group/role. The application level user-management managed user access to the specific application modules and associated queries (and data). It defined the manner in which each individual user accessed each module, defined the types of permissions (read or write/update) each user had on the data associated with each module, and defined the specific view of data for that user or the set of users. This was accomplished largely through the creation of user groups/roles (similar to the system level) with the exception of the MAS.

The MAS application system included 19 application modules. Each module performed several functions such as querying accounts payable, maintaining accounts payable master, identifying payment cycles, and reporting receipts that are not invoiced. These functions may be part of a single application module. As each module performed several functions ISI Inc. managed users by providing them access to specific functions within each module. To accomplish this, each module had several groups, each group with the privileges to execute a specific set of functions. A user was assigned to a group depending on the function(s) he/she needed access to. Each user was part of many different groups in different modules. The MAS hence had 277 unique user identifiers but over 9500 records to capture the user-access data to the different groups and application modules. The number of groups in each application module ranged between 6 and 50 (with an average of 27). There were over 500 such groups distributed over 19 modules. The number of users in each group varied between 1 and 210. There were 19 groups with more than 100 users in each, 24 groups with 50-99 users each, 191 groups with 10-49 users in each, and over 270 groups with less than 10 users in each group.

Users in the CAS were managed using 108 groups. There were 482 unique user identifiers created in the CAS belonging to the 108 groups defined. The user records in CAS numbered more than 2500 as each user was assigned to several groups. Interestingly, 16 of the 108 groups created did not have any users, 3 had over

380 users in each, 5 with 50-99 users in each, 34 with 10-49 users in each, and 50 groups with 1-10 users in each. The PDS was accessed by 19 unique users and managed with 5 groups (one with 10 users and the other 4 with 2 or 3 users in each) and each user belonged to one and only one group in this system. The CSS users were divided into three categories based on how they used the system: the developers, the database users, and the CSS application users. The 5 developers were assigned to one group and their privileges managed using that group. The database users (22 users) were managed using 3 groups. The administrator of the Oracle database was the only member of the administration group. Two experienced Oracle users were allowed to create ad-hoc queries from any of the tables in the database and were part of the second group. The rest of the database users belonged to the final group that permitted them to run pre-defined periodic reports. Members of the first and second group were also part of the third group. The CSS application users were managed using 10 groups. Some of these groups are customer support, sales, administrative groups for each, system administrator, training, and a group for supervisors. Of the 86 application users, 63 belonged to the customer support group.

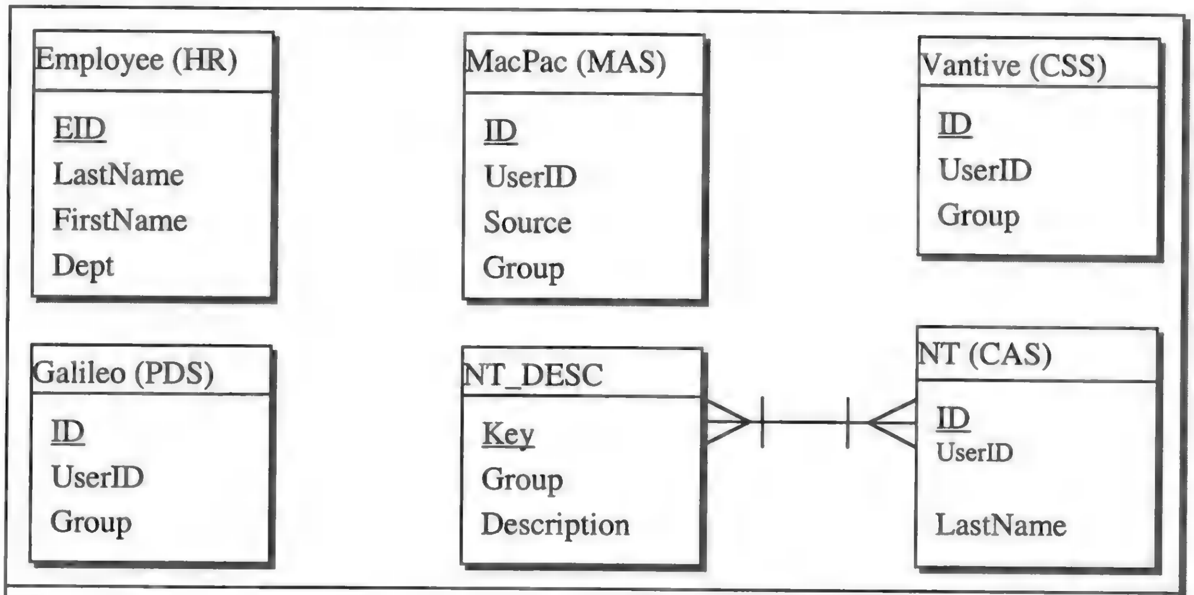


Figure 1 Original Schema

Employee information was maintained and managed by the HR department. Each employee of ISI Inc. was permitted to have multiple user identifiers and the first problem that ISI faced was matching employees with their corresponding user identifiers. Many of the user accounts were no longer used and these accounts continued to remain in the different systems at ISI Inc. Also, HR did not have employee information for many user accounts since some accounts were given to contracting employees who were not maintained in the HR database. These problems surfaced when the managers of functional units in ISI Inc. were asked to verify each of the user accounts and the access these accounts had on their systems. The purpose was to ensure that the right employees had the necessary access to the system functions and data and to remove the defunct or unauthorized accounts. No coherent report on users of systems could be generated using the existing data on user accounts. The management discovered that integrating the set of users across all of the systems in ISI Inc. was a complex task. ISI Inc. also understood that this situation compromised information security at ISI Inc. The specific objectives in integrating the user accounts included the following:

1. Reconciling employee names with user accounts to identify the employees (and contractors) who owned user accounts on the systems.

2. Identifying the groups on systems and modules in systems and understanding the privileges

given to each of these groups. Almost all systems had multiple distinct groups with the same set of privileges but were managed separately. Users were assigned to one or the other and some users belong to both groups.

3. Identifying users of each system and the groups within the system that the users belonged to. This aids in better managing each user with a single user account and reduces the "garbage" in the user management data.

4. Removing defunct accounts, and more importantly, removing accounts for employees who were no longer with ISI Inc.

5. Communicating to each user the specific manager(s) he/she should report to when accessing restricted data in each system.

6. Generating reports on users to assist managers in identifying and authorizing the users of the system that the manager was responsible for.

3. Designing a Database for Integrated User Management

The organization had four systems, each maintaining their own user accounts: a Windows NT 4.0 domain, and three mainframe systems: Macpac, Vantive, and Galileo. In addition, there was a database, unrelated to the user accounts databases on any of the systems, that stored employee information, such as employee IDs, names, and so forth. The schema we started with is depicted in Figure 1. The problem was that we had to answer the question: who has user accounts on which system, and what groups are they members of? None of the systems could provide us with that information, and of the four, only Windows NT stored anything that could be used to relate UserIDs to employee names.

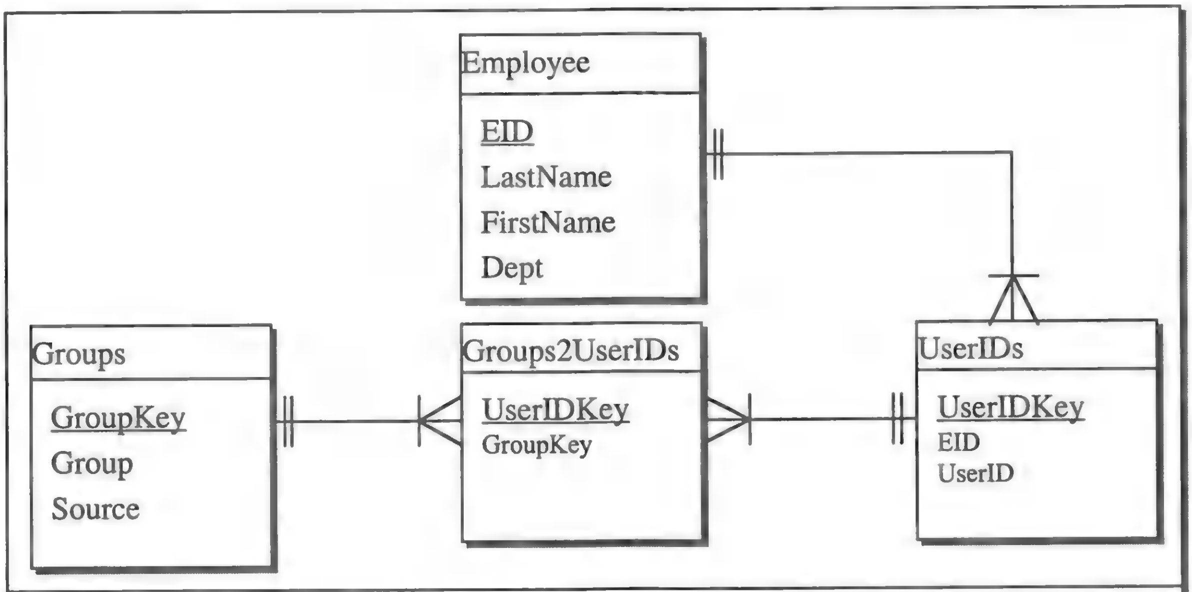


Figure 2 Optimal Schema

We started by defining the optimal schema; essentially stating what we wanted to know. Then the task was to fit the data we had into the schema that we wanted to produce the result we wanted: a list of employee names that listed their usernames and access to all systems.

The optimal schema is shown in Figure 2. The problem, however, was how to migrate the data from the original schema to the optimal one. The additional data in the Windows NT account database presented us with an avenue. We therefore used NT to create the link between the employee table and the UserIDs table.

The first step was to remove all duplicates from the NT table, and then join it onto the Employee table to map the NT usernames to employee IDs. This was done using a right join of Employee onto NT. The right join ensured that all NT usernames, even those that did not match an employee ended up in the table. The problem with this approach is obvious: Employee first and last names are very often not unique. To deal with that problem we put the data into a temporary table, called NT_Users, which we could analyze for duplicates later.

To match up usernames to employee IDs, a method for relating usernames to employee IDs is needed. With the Windows NT accounts, we can assume that if the first and last name of the NT user matches a first and last name in the Employee table, we can relate the NT username to the matching employee ID. However, the remainder of the systems do not have any name information, other than the UserID itself. We therefore need to make the following assumption:

Assumption 1: *If a username in any of the other systems matches a username in the Windows NT database, they refer to the same user.*

Assumption 1 sounds very logical. But, it is quite a leap of faith. However, in the situation where the systems have grown uncontrollably without a central authority, such as the one presented here, it is the best we can do. Since the employee database contained a manager for each employee, a business decision was made to create a list of user accounts for each manager and make the manager sign off on the list and report any anomalies, such as employees that no longer work for the organization.

Under assumption 1 we can now go ahead and match user accounts in the various systems. We do this by using a left outer join of the appropriate table with the temporary table created to hold the NT Users. The join condition is the UserID. This ensures that all those instances of users that do not match a UserID in the NT Users table get included. For those that do match the IDs in the NT Users table we also add the EID. Doing that for all the other systems results in a set of new tables that have the appropriate UserIDs for each user, as well as an EID where available.

Certain UserIDs did not match a known entry in the Employee table, and consequently no EID could be automatically assigned to those entries. A separate table called MISMATCH was created to highlight these entries. The MISMATCH table contained only three attributes: the UserID, the EID (which started out blank), and the source of the entry (the system it came from, such as NT, Macpac and so on).

The entries that end up in the MISMATCH table essentially fall into three categories:

1. Contractors who are not regular employees of the corporation and therefore do not have an EID, or even an entry in the Employee table

2. Names where the entry in the NT Users table do not match that in the Employee table, such as Jim versus James

3. Employees who changed their names for some reason, such as a marriage

Of these, numbers 2 and 3 are relatively easy to deal with by inspecting the entries and comparing them to entries with the same first or last name in the Employee table. This requires two queries, one that joins the MISMATCH table onto the employee table based on first name only, and one that joins the MISMATCH table onto the employee table based on last name only. Historical versions of employee tables help greatly in identifying entries falling into category 3. In the end, those entries that could not be resolved in this manner were passed to a manager to authorize. This was done for those entries where uncertainty remained as to the true owner of the UserID

Contractors presented a special challenge. These entries do not exist in the Employee table, for several reasons. Therefore, it becomes a challenge to locate the person and that person's manager to authorize the account. In section 5 some strategies for using Windows 2000's Active Directory for better tracking these users are presented. However, in most cases, it holds that it is imperative to track contractors in some form of database. This could be either the same or a similar format database as what is used to track employees. If contractors do not have an EID, one should be assigned to them that then links to the employee table such that a manager can be tracked for that contractor.

Once entries in the mismatch table have been assigned IDs, those entries can be re-introduced into the tables holding the users from the particular system.

After verifying that all UserIDs in the various user tables have EIDs, we can export them to the UserIDs table, and turn to dealing with the groups.

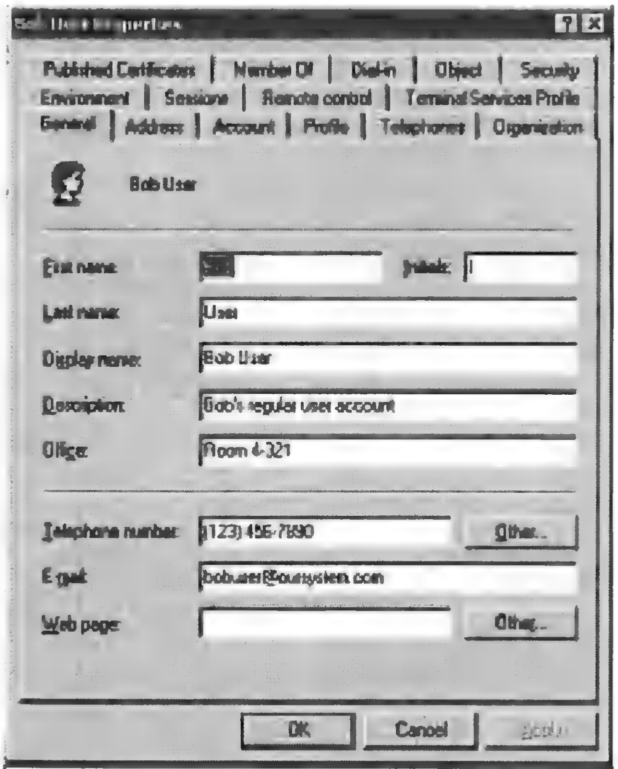
To build the Groups table is not terribly difficult, given that the original User tables have a group for each person. The problem is that the tables are not in any kind of normal form. Hence, if an employee is a member of several groups, that employee had several entries in the user table. A simple SELECT DISTINCT from each users table solved the problem and populated the Groups table with all the groups. As part of each SELECT query from the users table the source system for each group was populated as well, using a predetermined string.

At this point, three of the four tables in the final model are finished. The only one remaining is the intersection entity Groups2UserIDs, which holds the primary keys from the UserIDs and Groups tables and maps UserIDs to the groups they belong to. Creating that table, again, is not difficult. A single query can be constructed for each source (system) that joins the user IDs in the UserIDs table to the original IDs in the source table, extracts the group information, and inserts this into the Groups2UserIDs table.

4. Tracking Better Information

One of the major problems faced when attempting to validate users is that of incomplete information. Most systems do not allow you to track any information about user accounts, other than the account itself and any groups that the account is a member of. However, as we saw in the discussion above, this leads to problems when attempting to identify the users. Without a reliable way to tie a user account to a person, administrators cannot be sure whom the accounts belong to and who is responsible for the security and proper usage of that account. A very basic security requirement is that the system is capable of holding users accountable for their actions on the system. However, holding user accounts accountable is not useful. For accountability to be meaningful, we must be able to map user accounts to a physical person, which can be held accountable for the actions taken with that account. In addition, if a person no longer needs the account, we would need the person's manager to decide whether to disable the account. This information, as well, must be tracked by a system tracking user accounts. As we saw above, only Windows NT had some ability to track that information, in that it can track a person's name. However, even Windows NT fails in its ability to reliably connect that to an entry in an employee database. None of the other systems had even that basic functionality.

In February of 2000, Microsoft released the long awaited upgrade to Windows NT 4.0. Originally named Windows NT 5.0, the market name for the product at release is Windows 2000. Windows 2000 is quite possibly one of the most complex products ever designed by man, in any category. One of the features that Mi-



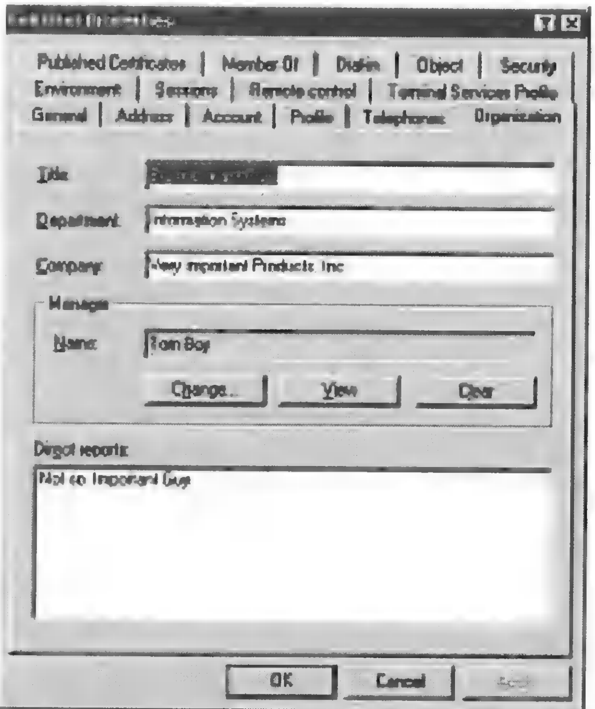
Bob User

First name: [illegible]
Initials: I
Last name: User
Display name: Bob User
Description: Bob's regular user account
Office: Room 4-321
Telephone number: (123) 456-7890
E-mail: bobuser@oursystem.com
Web page:

Figure 3 User Properties Dialog

crosoft added to Windows 2000 is the ability to track more information about users. Windows 2000 is built around a directory services module known as Active Directory. The Active Directory is essentially a database of objects that exist in an organizational computing environment. It can track not only user accounts but also computer accounts, groups of users and computers, organizational groups (known as Organizational Units) and many other objects. Microsoft's objective was to supplant the organizational employee database with Active Directory. Whether they have succeeded is a matter of significant debate. What is certain, however, is that the Active Directory can track much more information about each user than the SAM database in Windows NT 4.0 could.

Figure 3 shows the user properties dialog in the Active Directory. It is possible to track not only the user's name and a description, but also telephone number, office number and e-mail address. Furthermore, by clicking the Organization tab, one is presented with the screen shown in figure 4.


Published Certificates | Member Of | Dial-in | Object | Security
Environment | Sessions | Remote control | Terminal Services Profile
General | Address | Account | Profile | Telephones | Organization
OK
Cancel
Apply
Change...
View
Clear

Title: [illegible]

Department: Information Systems

Company: Very Important Products Inc

Manager

Name: Tom Boy

Direct reports:

Not so Important Guy

Figure 4 Organizational Information Tracking

Active Directory permits tracking of organizational attributes, such as the person's title, the department they are with, and the manager. These are all items that were stated as missing from the systems involved in the project described above, and are a significant advance. It is also possible to interface with the Active Directory through a set of APIs that can be used directly by a programming language, or through a data provider, such as the ActiveX Data Objects (ADO). This affords the ability to create an application that simultaneously queries a SQL database that serves as a repository for user accounts from a different system and the Active Directory, and presents this information in a uniform manner.

This means that the Active Directory can be conceived of as a node in a federated database system, and treated as a source by an application that provides services such as a repository of users. The reporting capabilities are extended significantly over what is possible with the information present in the systems described above. However, this information needs to be tracked in the Active Directory to be useful. To enable an organization to track that information, a procedure such as that outlined above is necessary to identify the existing user accounts. Few organizations have the luxury of starting over with their user accounts databases today.

There are several very important pieces of information missing in the Active Directory and assumptions that were made in its design. The first one is that an employee only reports to one manager. This is not necessarily true. Many organizations employ a matrix structure where people have multiple managers. In addition, certain employees are "lent" to other managers on a project-by-project basis. These employees may have a permanent manager and a temporary manager, each one of which might need to be notified of events regarding that employee. Furthermore, category three of the users found above to have no Employee ID were employees that had changed their names, e.g. as a result of a marriage. Therefore, it may be desirable to track employees' former names to be able to match user accounts on different systems to those on Windows 2000.

All of these values can be added to the tracking capabilities in Active Directory by means of extending the schema of the Active Directory. The Active Directory schema is designed such that it can be extended with new types of objects and attributes. This can be done through the capabilities of Active Directory itself. However, schema changes are irreversible operations in Active Directory. While new classes and attributes can be disabled, the changes cannot be undone. Therefore, it may be preferable to create a middleware application that joins the Active Directory to a database that contains the additional attributes. Further research is needed into this technique to determine what is the best course of action. Depending on the operations needed, schema extensions may be the only viable solutions.

5. Conclusion and Further Work

In this paper we have described a typical problem that organizations face - that of managing user accounts belonging to the many different IT systems in the organization in an integrated fashion. Organizations are increasingly forced to deal with this situation due to the emergence of Intranets / Extranets, the need to share data/applications both within and across organizations, and the need to migrate to new systems combined with the desire to track more information about users on those new systems. Ad hoc creation and management of user accounts to support Intra/Extranets results in the wasteful duplication of user information. More importantly, managers lack the ability to identify and associate employees with user accounts and match user accounts with system/application privileges granted to each. This compromises the security of their corporate network of systems and must be resolved. In this paper we have examined one such situation and proposed a practical implementation to address the resulting problems. The proposed solution offers several benefits besides being simple and practical. It is not specific to managing users on a particular system and can be used in an integrated environment with several types of systems as illustrated here. Further, it is scalable to include applications and networks that users may be authorized to access and hence requires user management. Finally, it can be implemented in any relational database system or incorporated into the Active Directory of MS-Windows 2000 to exploit the user accounts and related user information that is already tracked by that database.

In extending this work, there are two specific research directions that need to be investigated. The first is to examine the capabilities of the Active Directory with the view to extend our solution to include the capabilities in this user management tool. Active Directory performs extensive user management services but is restricted to the Windows 2000 network. Organizations typically maintain several networks of systems and managing all of these using Active Directory is impossible. The current solution described here has a very limited set of user management capabilities when compared with those offered by the Active Directory. The objective of this extended solution is to provide organizations with the ability to comprehensively (all user management services) manage user accounts across all networks in an integrated fashion. An application that integrates the Active Directory with the user account databases of several other systems is therefore needed. This application could store additional data in a database of the administrator's choice, and have the ability to act as the interface to all of the managed systems. Such an application is under investigation by the authors of this paper.

Each individual operating system (network or otherwise) and application supports the management of users on that system/application. One approach to integrating several such systems/applications/networks for integrated user management is to model user information in each system as a simple relational schema. The resulting set of schemas can be integrated to create a federated/global schema by applying well-known techniques for schema integration. This type of schema is a requirement for the management system described above. Several questions need to be answered before realizing this solution. Would the federated schema accurately track user account information? How scalable or extendible is this schema and how would changes to individual schemas be reflected and managed? What is the overhead cost/time involved in creating and managing such a federated schema? Attempting to answer such questions and defining a global solution for integrated user management is the second research direction that is currently under investigation.

References

Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria," Report DOD 5200.28-STD, 1985.

Remote Windows NT Administration Using Windows CE Handheld Devices

Craig Stacey
Argonne National Laboratory

Abstract

As a systems administrator on call at any time of day, I want to explore ways to manage the environment and handle emergencies remotely. In this paper, I look at a number of options for remote administration using compact devices. While the most obvious form of remote administration is a laptop computer, not every organization has the budget to provide these to their systems administrators. Also, in some situations, even a laptop computer is too large to carry around conveniently. Therefore, I evaluated three devices to determine their usefulness for remote administration. While concentrating on Windows CE devices, I included a Palm Computing handheld for comparison purposes.

1. Overview of MCS environment

The computing environment in the Mathematics and Computer Science Division of Argonne National Laboratory consists of nearly a thousand computers, including supercomputers, servers, workstations, desktop machines, and laptops. Our Windows infrastructure consists of 13 production NT servers, eight Samba servers, eight experimental NT cluster servers, approximately 90 NT/2000 workstations, and 30 or so machines running Win95/98/2000 (mostly laptops).

The systems group that manages these machines consists of eight full time administrators, three of whom are at least partially responsible for the Windows environment. As with all other systems administrators, I have plenty to do, and our user expectations are high. This paper will deal with an experiment in remote administration: namely, using Windows CE devices to remotely administer our Windows servers.

2. Why use a handheld?

I did this experiment for a couple of reasons. First, it seemed like a great academic exercise. There's a certain challenge that comes from administering your organization's web server or Primary Domain Controller via a small handheld computer. However, there's also the allure of being truly able to administer servers remotely without having to carry around a laptop.

The advantage of such a model, should it prove successful, is an ability to carry a pocket-sized device and yet have full remote terminal capabilities. You'd not have to leave other situations for some emergencies, instead using your handheld device to solve the problem over a phone line, Ethernet, or wireless link. The usefulness of this becomes very apparent when you're at some social event or in a meeting.

It was only recently that our division's budget was large enough to supply the systems administrators with laptop computers. The cost of a Windows CE device is much lower than that of a notebook, and may be more palatable to some organizations.

3. The Experiment

There are a number of things to consider in this support model. My focus in this experiment was on security, functionality, usability, portability, and efficiency.

For each device tested, I performed a series of tests to try to accomplish common administration tasks, plus some emergency procedures. The following are the functions I tried to do with each handheld:

- Change user password
- Disable/Enable user account
- Change file ACLs
- Stop / Start web services
- Reboot server
- Powercycle server
- Send and receive email (of course)

As most NT Administrators are aware, most of these tasks can be accomplished from the commandline, but I wanted to be able to access the GUI as well, in case something went wrong with the telnet daemon.

3.1 The tools of remote systems administration

SSH: Ultimately, SSH is a necessity. In the case of our environment, unencrypted logins are rejected at our gateway router, so without a method of secure login, nothing can be done offsite. Therefore, for a device to be considered a viable option, SSH must be available.

VNC: In the GUI-intensive world of Windows, Virtual Network Computing (VNC) is a godsend. Allowing remote control of the desktop via the GUI is a tool I've come to rely on, and is, for many tasks, a vital part of remote administration.

Windows Terminal Server: When possible, the use of Windows Terminal Server is preferred to VNC. It's much faster, and does not have the SSH overhead for security.

SSL capable web browser: For some functions, a web browser is enough to accomplish some administrative tasks, thanks to some tools we employ at MCS, as well as Microsoft-supplied tools.

Ataman Telnet/RSH daemon: We've been very happy with this software package. While telnet, rlogin and rsh are blocked at our gateway router, they are allowed within our switched network. This allows us to integrate our Windows environment into parts of our automated unix administration tasks.

Perl and NT Resource Kit: Of course, perl is used extensively on our servers for administration. Combined with the Windows NT Resource Kit tools, a set of commandline tools exist for controlling and administering our servers, which can be quite desirable over a slow phone link.

Baytech Power strips: These remotely controlled power strips are wonderful. If a server is wedged beyond hope, being able to power cycle it remotely is quite helpful. You need to confirm the PC will come back after a hard power cycle. On some ATX motherboards, this is simply a BIOS setting. On some of our machines, though, I discovered I couldn't make the machine automatically come back, so I had to enable and use Wake-On-LAN.

3.2 The devices I tried

I should note that just as this paper was finished, Microsoft unveiled the next generation of Windows CE device, called PocketPC. Unfortunately, as of this writing, I've yet to experiment with these devices, so it's possible some of the statements I make do not apply to this new version. I've noted in the device listings which version of the operating system they are running.

I've been testing with two different Windows CE devices (one handheld, one palm sized) using a variety of connection methods including 56K modem, 802.11 wireless, 10BaseT Ethernet, and CDPD wireless modem. I have also, for the sake of completeness, looked at options under the Palm Computing platform using a Palm III.

The Computers:

The Handheld PC ("H/PC") is an NEC MobilePro 800. The device has 32MB RAM, a MIPS 4000 processor, built-in 56K modem, USB, IR port, one Type II PCMCIA and one CompactFlash PC Card slot. It runs Windows CE, Handheld PC Edition Version 3.0.

The Palm-sized PC ("P/PC") I tested is a Casio Cassiopeia E105. The device has 32MB RAM, a MIPS R4000 processor, IR Port, 65K color, and one CompactFlash Type II slot.

The Palm Computing Platform device used was a 3Com Palm III running PalmOS 3.1.

The Accessories:

I used Aironet 802.11b (aka 802.11 High Rate) wireless cards and access points to test the wireless connectivity. These were ideal as they were one of the first to market with an 11Mbit 802.11 standard wireless card, and had Windows CE drivers available right away. Preliminary tests with Lucent WaveLAN cards and drivers were problematic, but it's entirely possible these issues have been resolved. 802.11 Wireless cards are, at present, only available as a PCMCIA Type II card, and therefore will not work with the P/PC.

For wired ethernet, I used Socket Communications ethernet cards. These were the best card to use due to the fact they were designed with Windows CE in mind and have a very low power consumption. One of the cards tested was a CompactFlash ethernet, which meant I could use it with the P/PC.

The H/PC had its own built-in 56K modem. For the P/PC, I used the Casio 56K CompactFlash card modem designed for the E105. A standard Palm modem was used for the Palm III.

For wide-area wireless, a Novatel Wireless SAGE CDPD modem was used. This is an external device with a serial connection, allowing it to be used with all of the tested devices. CDPD connectivity is available in most major cities in the US. It has a bandwidth of 9600 bps.

3.3 Keeping it secure

We have a policy within our organization that states "External network connections to MCS resources shall not use clear text reusable passwords for authentication." In enforcing this policy, our gateway router rejects well-known services that use cleartext passwords, including telnet, POP, IMAP, and FTP. In addition, because we won't trust rlogin and rsh connections from offsite, and because additional authentication is sent cleartext, we block those protocols as well. Therefore, our only real method of performing tasks in any of the above protocols involves Secure Sockets Layer (SSL) or Secure Shell (SSH).

Our policy defines an external network connection as one that originates outside our gateway routers. We trust that our internal network is secured, as most traffic inside is sent over a switched network, making packet sniffing ineffective. We do still encourage encryption where possible, but it is not required. Our dialup connections originate outside the router, technically, but because the first point on the router is an address inside our network, we treat these dialups as if they are secured.

4. The Results

H/PC (NEC MobilePro 800):

Because this has both a CF card slot and a Type II PC card slot, I could test every connection method, and all were solid. The built-in modem was quite convenient, and I found I often left the wireless card in the unit rather than tether it with a wire. This did consume more power, but with no hard disk, the battery life of this notebook-sized device was still quite impressive.

The SSH client for Windows CE I used is SSHCE, a product currently in beta testing written by mov software (http://www.movsoftware.com). It's not the best SSH client I've used, but it's the only one for Windows CE, and it allowed me to login from offsite. From that remote login, I was able to reboot or powercycle the server, change user passwords, edit various configuration files, make registry changes, and use other administration tools all from the commandline.

VNC for Windows CE works surprisingly fast, and the 800x600 display on this handheld was quite nice for this. Double-clicking with a stylus is an art to be mastered, but I found my work with this, and with the other handhelds, was improved by the use of a fingertip stylus. All in all, once a machine is up and running, VNC was a good option. Technically, VNC meets our secure access policy, as the challenge and response are encrypted. However, once the session is active, all data is sent cleartext, thus any passwords I type in that VNC session are snoopable. Therefore, we at MCS recommend to our users that any VNC sessions be tunnelled through SSH. Unfortunately, SSHCE does not support port forwarding as of this writing, so that's not an option. I therefore limited our VNC experiments to onsite wireless, or offsite dialup.

What blew me away was Windows Terminal Server Client for Windows CE. This was fast and a joy to work with. I'd often forget I was using a WinCE device, as it looked and felt like a Windows 2000 machine. This was, by far, the most usable method, and I even relied on it for a couple of weeks while my laptop was being repaired.

By using the web-based administration feature of IIS, plus a handy user administration tool written by Mike Gomberg, I was also able to accomplish certain tasks using a browser, such as web service stop and start, password changes and account activation/deactivation.

Impressions:

Ultimately, this was the most usable of the devices. Not surprisingly, it is also the largest of the devices. In fact, there are now a number of micro notebooks running Windows 98 or 2000 that are smaller than this. It does have a standard laptop beat in terms of weight and battery life, as well as overall price, but if you're committing to carrying something of this size, you might be better served by getting a micro notebook like a Sony Vaio or an IBM Thinkpad 240.

P/PC (Casio Cassiopeia E-105)

I had the most hope for this device. It's compact, has a fantastic battery life, and a nice display. Unfortunately, the people writing tools for Windows CE aren't concentrating on the P/PC line. It seems a lot of the programs available for this are games and entertainment oriented. At the time of writing, the only version of SSHCE I received was written only for the H/PC. A P/PC and PocketPC version is now available, and I've requested a copy of the beta. At the time of the experimentation, however, there were no SSH clients available. Aside from SSH, another stumbling block was the lack of a Windows Terminal Client for the P/PC. The absence of an encrypted connection method is is problematic, but not insurmountable. It simply requires I use a secure dialup for any offsite activity. This means the Sage CDPD modem was not really usable for this device.

I don't view this as a great setback, as the Sage modem is larger than the P/PC itself, and part of the appeal of this small device is that it's something you could conceivably have with you at all times.

VNC for Windows CE works, although it's obviously not designed for a screen that small; you can't see the entire hostname dialog box, for example, but if you trust your stylus typing or Jot writing, you can connect. It's pretty slow, especially over a dialup. I'd say it's something mostly useful in emergency situations, but not something you want to use every day.

TelnetCE from ActiveBridge has been working quite well, and I was able to use it over our secure phone link. This allowed me access to the commandline tools I like to use in administering the servers.

I'm not sure why Microsoft chose not to install a web browser on the P/PC devices, and I'm sure this has been rectified with the PocketPC. IBrowser, a free program from Foliage software (http://www.foliage.com) can perform most web browsing tasks, but it does not include SSL capability. They offer a $34.99 upgrade to Ibrowser Plus which allows https connections to work, and with that tool, the same web-based admin tasks can be accomplished on the P/PC as with the H/PC.

Impressions:

This was, by far, the most promising device. Compact enough to carry with me wherever I went, this could be the device that would get me out of having to make the long drive to work if something went wrong after hours. The battery life is incredible, and with the release of the PocketPC platform, it's possible more serious development will happen for these devices.

Alas, the lack of a Terminal Server client for the P/PC hurts. Were this tool available, the P/PC could be so much more useful as a remote administration tool. As it is, though, it's a handy device. The model I tested did not have a built-in modem, so a modem CF card and dongle were required. This is not a lot of extra baggage, and indeed the P/PC is the smallest of the configurations I tested. I look forward to trying out the new PocketPCs, in the hopes that these are even more useful.

Palm Computing (Palm III)

In an effort to make sure I was fair and evenhanded, I tried using a Palm III for various administrative tasks. A color palm was not available during testing, so the lack of color on the screen, while certainly annoying in VNC, is not anything you can't get around if you need to.

Top Gun SSH (http://www.isaac.cs.berkeley.edu/pilot/) for the Palm III is not bad. Certainly, it's no worse than SSHCE in terms of speed and screen refreshes. Like SSHCE, it lacks port forwarding, so using VNC is restricted to secure phone line use. This isn't especially problematic, as our testing of the Palm III involved only Palm Modem tests. PalmVNC was a bit slow and buggy, and I didn't find it terribly useful, although it would certainly work in a pinch.

ProxiWeb by ProxiNet (http://www.proxinet.com) is a free SSL capable browser for Palm III. Using this, I could accomplish all the same web-based admin tasks as I could on the P/PC and H/PC.

Impressions:

The Palm is usable at the same fundamental level as the other two devices. Ultimately, if I can use the device to get a unix shell on one of our workstations at MCS, I can accomplish any of the emergency tasks I'd need to, and many of the day-to-day tasks. The Palm III certainly allows me to accomplish that. With the advent of the wireless Palm VII, I'm sure it becomes a much more viable remote administration tool. I look forward to experimenting with that model.

4.1 What worked well

On each of these devices, I was able to gain access to a command shell, which, as stated before, allowed me to accomplish just about everything I needed to. A good knowledge of the tools available for the commandline is handy. An added benefit of using these devices is also storing references and documentation on them.

They all did a passable job of running VNC, allowing me to perform GUI intensive tasks, including restarting a wedged telnet daemon.

Secure web-based administration went fine, although in the case of the P/PC it required the purchase of a product that really should be included.

Terminal Server Client is a wonderful program, and turned the H/PC into a full powered notebook computer. It's fast, efficient and secure.

4.2 What's missing

Obviously, Terminal Server Client would have been quite handy on either the P/PC or the Palm III. I hope Microsoft will at least develop a client for the PocketPC. Such a move could turn the PocketPC from a neat toy to a vital systems administration tool.

A port-forwarding SSH client would also be useful for those of us who work on sites that are paranoid about security.

5. Conclusions

Wireless internet access is becoming very common, and it would be nice for systems administrators to have access to the right devices and software to maintain their machines remotely. I hope developers at Microsoft and their hardware partners are envisioning the PocketPC to evolve into a wireless Windows terminal. Were that to happen, I can't imagine anyone who does this for a living not having one at his or her side. We're not there yet, but we could get there in the very near future. With the tools available today, you're essentially a phone call away from your machines at any time.

It's an interesting support model, and one that I think other systems administrators should think about. It's not right for every situation, but it's proven useful on more than one occasion, and is especially helpful in emergencies.

6. Author and Project Information

Craig Stacey is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. His email address is stace@mcs.anl.gov.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Kerberos interoperability issues

Paul B. Hill
Massachusetts Institute of Technology

Abstract

MIT's computing environment is a heterogeneous environment that has used Kerberos as a primary authentication method for over a decade. Instead of migrating our existing KDCs to Windows 2000 we have chosen to use cross realm trust to support our Windows 2000 computing environment. During our deployment project we have encountered some interoperability problems and have worked with Microsoft to resolve these. We have also encountered protocol extensions that have been used by Microsoft and we have been working with Microsoft under the umbrella of the IETF to have these documented. Some of the problems were only identified and resolved by analyzing network traffic.

1. An Introduction to Kerberos

Kerberos is a secure authentication protocol for use in distributed computing environments. When used ubiquitously within a computing environment it can also provide single sign-on capabilities. Kerberos was first introduced to the USENIX community in 1988 in the *Proceedings of the Winter 1988 Usenix Conference* and *Proceedings of the Usenix Workshop on Workstation Security*. Articles on Microsoft's implementation of Kerberos have also appeared in the November 1997 and May 1998 issues of *;login.*

The protocol was initially developed at MIT as part of Project Athena. The work was funded by IBM and Digital. The source code was made available to others and the copyright allows the development of derivative and commercial work with few restrictions.

MIT remains very active in the development of Kerberos. Change control of the protocol is now the responsibility of the IETF. Version 5 of the protocol is currently defined by RFC 1510. Revisions to the protocol are in progress and there are related drafts closely associated with Kerberos being developed as well.

MIT had several goals when developing Kerberos. These included:

- raising the awareness of security issues in a distributed computing environment, especially those that rely on packet technology
- providing a solution to security problems that nobody else was attempting to address at the time
- encouraging vendors to adopt Kerberos so the we could purchase secure systems for our own needs

Although this paper does not explore many details of the protocol, some basic definitions will be helpful for understanding.

User - A human being who wishes to a use a computer system.

Service - An abstract specification of some actions to be performed. The actions are performed by a program or set of programs running on a computer which is accessible over the network.

Principal - An entity which can both prove its identity and verify the identities of other principals who wish to communicate with it; each **user** and each **service** registered with Kerberos is thus a principal.

Ticket - A block of data which, when given to a user, enables him to prove his identity to a service.

Realm - an authentication domain or authentication namespace.

TGT - A ticket granting ticket. A special Kerberos ticket which enable a user to get other service specific tickets.

KDC - Key Distribution Center. The Kerberos server that provides tickets to users.

1.1. MIT's use of Kerberos v4 and v5

MIT uses Kerberos throughout its academic and administrative computing environments. Our computing environment is heterogeneous and includes many versions of UNIX, UNIX variants, IBM's VM, Apple Macintosh OS, and most versions of Microsoft operating systems.

Although Kerberos v4 and v5 do not interoperate both version 4 and version 5 of the Kerberos protocol are used at MIT at this time. We expect this situation to exist for many years. Support of both protocols may be achieved by one of two mechanisms. The MIT Kerberos 5 release can speak the Kerberos 4 protocol, assuming it was built with the "`--with-krb4`" option (which is the default). Or separate KDCs can be maintained for each protocol; the database information may be propagated between the different KDCs to simplify administration.

The applications used consist of a mixture of internally developed applications and commercially available packages. Examples of the commercially available packages include Transarc's AFS, which uses Kerberos version 4, and SAP R3, which uses Kerberos version 5.

Despite supporting both version 4 and version 5 of the protocol the MIT Kerberos environment is remarkably simple. MIT does not use DCE anywhere in its infrastructure. At this time only a single Kerberos realm is used on campus for all supported services. This is the ATHENA.MIT.EDU realm. Although many other realms exist on campus they have no operational impact beyond a very small set of users. They only exist for testing, research, or as educational exercises. Cross realm trust relationships are not established with departmental Kerberos realms.

The Athena realm is maintained by Information Systems and of course runs the MIT implementation of Kerberos.

1.2. Microsoft's support of v5

The default authentication protocol used by Microsoft's Windows 2000 operating system is Kerberos version 5. The Kerberos protocol is just one of the security protocols supported by the operating system. Others include NTLM for backwards compatibility, SSL / TLS for public-key authentication, SPNEGO for security protocol negotiation, and IP Security (IPSec) for network layer security.

Windows 2000 only supports Kerberos version 5. There is no support for Kerberos version 4, nor is DCE style cross-realm trust supported. If an organization requires Kerberos version 4, or DCE security, support the organization must examine its interoperability options and develop a strategy.

Microsoft provides and uses a Security Support Provider Interface (SSPI) to the Kerberos protocol. The Kerberos security provider may be used by any application designed to use SSPI for network security. Microsoft is using this support to secure extensive portions of any Windows 2000 infrastructure. A number of services support Kerberos authentication in Windows 2000. Here is a partial list:

- authentication to the Active Directory using LDAP for queries or directory management
- CIFS/SMB remote file access protocol
- distributed filesystem management and referrals
- secure DNS address update
- print spooler services
- optional IPSec host-to-host authentication in ISAKMP/Oakley
- reservation requests for network Quality of Service
- intranet authentication to Internet Information Server
- authentication of public-key certificate requests to the Microsoft Certificate Server for domain users and computers
- remote server or workstation management using authenticated RPC and DCOM

The support of Kerberos over such a wide range of systems does imply that Microsoft has a high level of commitment to the SSPI interface and that this is a flexible interface. The SSPI interface is very similar to the IETF's GSS API, RFC 2743. Microsoft has stated that in order to have all of these services support Kerberos the most complex change was to the SMB server, which did not use SSPI prior to Windows 2000.

1.3 Microsoft's Windows 2000 Kerberos implementation

There are several more important aspects to Microsoft's Kerberos implementation that system architects should keep in mind.

Every Windows 2000 Domain Controller is a KDC. The KDC is a logical process that is part of the LSA process. It cannot be removed.

Active Directory, via LDAP, is the administrative interface to the KDC. Please note that the administrative interface to the KDC has never been standardized through the IETF.

Although not required by the Kerberos standard, Microsoft's implementation requires that the DNS domain and Kerberos realm names be identical. Per the current standards DNS names are case insensitive and the Kerberos realm name will always be upper case. In the context of Windows 2000, a Domain encompasses both the DNS domain and the Kerberos realm. This overloading of terms can become very confusing when talking about configuration and support issues.

The Microsoft libraries locate the KDC using DNS service location records instead of relying on local configuration files. There is an IETF draft in progress to standardize this behavior. When using non-Microsoft realms for authentication local client configuration information is also supported.

To foster interoperability Microsoft implements DES-CBC-CRC and DES-CBC-MD5 encryption types. These are 56 bit symmetric key algorithms that are used by other Kerberos implementations. The implementation does not support the MD4 checksum type. Microsoft's preferred encryption type is RC4-HMAC. This is currently documented in an informational IETF draft.

Microsoft's use of a new encryption type had two motivations. Most importantly upon upgrading from a Windows NT 4.0 environment to Windows 2000, accounts will not have the appropriate DES keying material to do the standard DES encryption. If a new encryption type was not supported an organization would have to reissue passwords for all user accounts. In large environments this would be unacceptable. The new encryption type also helped Microsoft resolve some of the possible barriers to export of the software imposed by US regulations. Early in the development process, 56 bit DES encryption could not be exported.

Microsoft uses structured service naming conventions. This does raises some issues for developers wishing to use the GSS API libraries to support multiple operating systems.

Microsoft's implementation of the Kerberos protocol supports and assumes the use of authorization data in tickets. This is compliant with the current proposed draft revisions to Kerberos that the IETF is working on. Other implementations that include authorization data within the Kerberos tickets are DCE and Sesame.

The Windows 2000 authorization data is ignored by current UNIX implementations. Although Microsoft has released information about their use of the authorization field it appears that the Kerberos community is precluded from writing any code that can use this information in any way.

The Windows 2000 KDC supplies the authorization data that is placed into the tickets. Depending on the type of ticket the authorization data included may consist of user SIDs, global or universal group SIDs, or domain local group SIDs.

Application services that receive a ticket are able to extract the list of SIDs and use this information to determine what the client is allowed to do based on the Windows 2000 group membership information.

2.0 Interoperability scenarios

When we talk about interoperability we will use Domain to mean a Microsoft Windows 2000 Domain which by definition include a Kerberos realm. We will use the term realm to mean a Kerberos realm that is not a Microsoft Domain.

There are several interoperability scenarios that could be considered, not all are listed here.

- Windows 2000 domain without a Microsoft KDC
- Kerberos clients in a Windows 2000 domain
- Kerberos application servers in a Windows 2000 domain
- Standalone Windows 2000 systems in a Kerberos realm
- Using a Kerberos realm as a resource realm
- Using a Kerberos realm as an account domain.

The first scenario is not supported by Microsoft and not available to anybody at this time. Providing this option would require a 3rd party Domain Controller replacement or functionally equivalent clone. It appears likely that Microsoft will try to prevent any third party from implementing a solution that enables a customer to choose this option.

The Windows 2000 domain security model depends on the authorization information being present in the ticket. Microsoft is asserting intellectual property issues on their use of the authentication field and apparently preventing others from developing compatible implementations. Furthermore, the Microsoft KDC is tightly integrated into the Active Directory and LSA process.

2.1 Kerberos clients in a Windows 2000 Domain

Some organizations may find this option attractive. Suppose for example that an organization has a heterogeneous computing environment but does not use Kerberos today. If Windows 2000 is used for account management and authentication its use can be leveraged to improve the security of the other computing platforms as well.

Kerberos version 5 client libraries and applications are available for most versions of UNIX, Linux, and Mac OS. The versions of kinit, klist, kdestroy as well as other applications from the MIT distribution have been tested against the Microsoft KDC. No code changes were required in order to make the applications work.

The MIT Kerberos libraries will ignore the contents of the Microsoft authorization field, per the specification. The other operating systems would not be able to use the Microsoft SIDs to determine the intended authorization access. The authorizations methods supported by most UNIX-based services are application specific today.

Applications that use the GSS API and the Kerberos v5 mechanism, will also continue to function is this type of deployment.

Note that this scenario will not be suitable for organizations that intend to use DCE security for application services. The DCE libraries will be expecting different authorization data within the field and will not be able to use the data supplied by the Microsoft KDC. Nor is this scenario appropriate for organizations that need to support version 4 of the Kerberos protocol.

2.2 Kerberos application servers in Windows 2000 Domain

Everything that was stated in the preceding section applies to this scenario as well. There are also other situations where this configuration becomes attractive.

Suppose that you have a UNIX database server that supports Kerberos authentication and you would like to provide users within your Domain access to the database via a Web interface. Since Internet Explorer and IIS support authentication using SSPI it is possible to create a multi-tier application that uses Kerberos authentication.

You have to create and manage service accounts for the UNIX servers. In this case the computer accounts are the same as Windows 2000 user accounts. You may find it useful to create a separate organizational unit (OU) within the AD for these accounts.

You also have to create and install a keytab file on the application server. Microsoft provides the Ktpass.exe program as part of the Windows 2000 Resource Kit. This program can be used to generate the keytab file. You will have to copy this onto the correct host and merge it into the UNIX keytab file. Be sure to copy the file from the location where it was created to its destination in a secure manner.

Microsoft has published a Kerberos interoperability paper that describes the creation of the computer accounts and use of the keytab program quite thoroughly.

Note that in the case of a multi-tier application using IIS, the IIS might be trusted for delegation. This means that you could create a system that would use Kerberos authentication across all of the tiers.

2.3 Standalone Windows 2000 systems in Kerberos realm

This scenario is similar to that used by most UNIX centric Kerberos deployments today. If you have an existing Kerberos realm that provides application services and all network resources it may be attractive to you. It does assume that you plan to offer no Microsoft application services or network resources that support Kerberos authentication.

The Windows 2000 computers will not be members of a Microsoft domain. Local accounts will be used on each machine to establish an account mapping but authentication will be performed using a KDC that is not implemented by Microsoft.

Microsoft provides a Ksetup.exe utility as part of the Windows 2000 Resource Kit. This utility can be used to configure the realm information on each computer. The same program is also used to establish the local account mapping.

The local account mapping can be done on an individual basis where each account in the realm is mapped to a corresponding local account on the machine. This does not scale well.

An alternative is to map multiple individual accounts in the realm to a single account on the local machine. This may not be suitable for many environments.

2.4 Using a Kerberos realm as resource domain

Many sites have multiple user namespaces today. By this I mean that they manage user accounts for their UNIX operating systems independently from their user account management on NT, or other operating systems. If this works well for an organization the practice may continue after Windows 2000 is deployed. Despite this bifurcation there may still be a desire to provide services across the environments.

Suppose that one user population primarily uses Windows 2000 file and print services but many of the users also need to access some Kerberized services located on UNIX servers in a Kerberos realm, for example an IMAP service that supports v5.

By establishing a one way trust relationship between the Windows 2000 Domain and the Kerberos realm, such that the Kerberos realm trusts the Windows Domain, an organization can provide their Windows 2000 users access to the UNIX hosted IMAP service.

Users will initially authenticate to the Windows 2000 Domain Controller. When the SSPI enabled application needs to authenticate to the IMAP server the SSPI libraries will transparently perform the cross realm authentication and present the correct ticket to the IMAP server.

The application services in the UNIX realm will have to determine the access rights of the user. This will not use the Windows 2000 authorization data, instead the application server will resort to the methods that it normally uses.

If the organization wished to migrate to a uniform name space this strategy would still be useful. Over time the UNIX user accounts could all be migrated to the Windows 2000 Domain and the UNIX realm would only contain the service principal names for the UNIX based application services.

2.5 Using a Kerberos realm as an account domain

Now I'll focus on the deployment scenario that has occupied a great deal of time, and the time of John Brezak, Program Manager of Kerberos at Microsoft. I'll describe more of the details than have been covered in the other scenarios.

In this case the Kerberos realm will be used as an account domain and the Windows 2000 Domain will be used primarily to provide authorization data. This means that all of our users will initially authenticate to our existing Kerberos realm but we will still have an operational Windows Domain and all the services that it can provide.

Earlier I stated that MIT has only a single Kerberos realm that is used on campus for all supported services, and that cross realm trust relationships are not established with departmental Kerberos realms. Our Windows 2000 deployment is compelling us to change that slightly. We are creating a second Kerberos realm, in this case a Domain, and establishing a trust relationship with it. We are not going to make this a common practice since we wish to avoid a proliferation of realms that add no value to the community.

Our existing supported realm is ATHENA.MIT.EDU. It is used for authentication within the MIT.EDU DNS domain and the few DNS subdomains that we have.

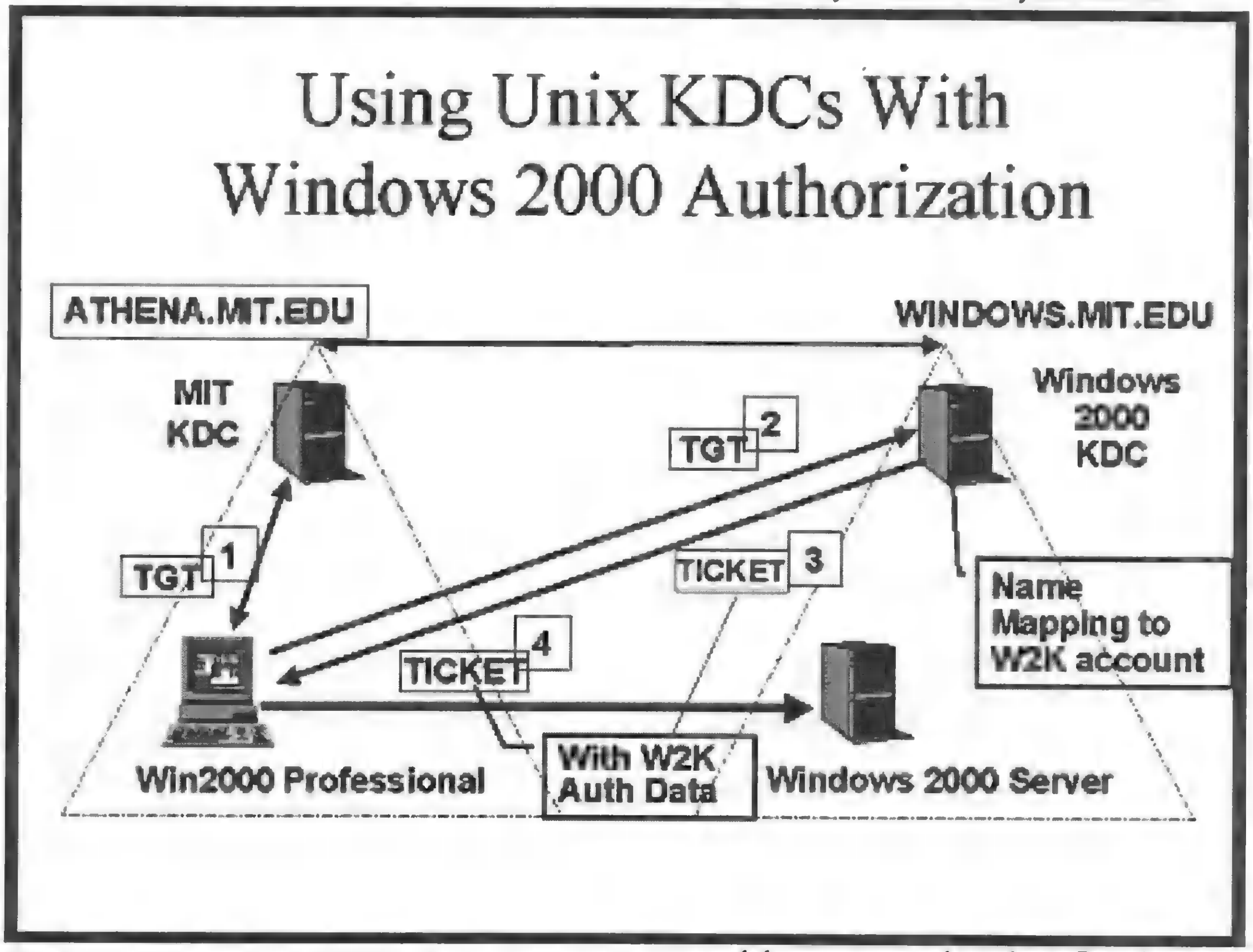
We have created a WINDOWS.MIT.EDU Domain, which means that we now have a WINDOWS.MIT.EDU Kerberos realm and a WINDOWS.MIT.EDU DNS subdomain. We will not be providing dynamic DNS for our domain, and we are not placing all Windows 2000 machines into the windows subdomain. The only machines that will actually appear in our windows subdomain are the Domain Controllers, our RIS servers, and a few other miscellaneous servers. All of the DNS information for the Windows 2000 workstations will reflect that they are in the top level MIT.EDU DNS domain.

We then created a trust relationship between the realms so that the Windows Domain trusts the Athena realm. In order to make the trust relationship useful, each Kerberos principal in the Athena realm has a corresponding account in the Windows Domain. This is done using the altSecurityIdentities attribute within AD. Each Win2K account has this attribute populated with the corresponding Athena principal information. For example, the Win2K account pbh@WINDOWS.MIT.EDU has an altSecurityIdentities entry that contains Kerberos:pbh@ATHENA.MIT.EDU.

This configuration enables me to log into a computer that is a member of the Windows Domain as pbh@ATHENA.MIT.EDU. I will obtain my Athena TGT so that I can subsequently access resources in the Athena realm. I will also automatically obtain as needed a TGT and service tickets within the Windows Domain. The Windows tickets will include my authorization data which domain applications can evaluate to determine my access rights to resources.

With the configuration and application requirements that have described so far there is no need to synchronize passwords between the Domain and the realm. A compromise to the security of the Win2K domain will not compromise any of our existing resources in the Athena realm.

The included figure shows a schematic representation the traffic that occurs when using this configuration.



2.5.1 "unknown passwords" in the Windows 2000 domain

Section 1.2 listed many Windows 2000 services that support Kerberos authentication, however not every Microsoft service or application supports SSPI today. For example, the Macintosh file services and the Macinstosh UAM.

If an organization needs to support the Macintosh file services within a Windows 2000 Domain, NTLM authentication will be used when accessing these services. A similar situation may exist when supporting "downlevel" clients, or application servers that do not use SSPI.

You have a couple of choices at this point. You can maintain separate passwords for your Domain and realm accounts, and hope that your users will not manually synchronize them. When doing this most applications will prompt the user for the Domain password when resorting to the NTLM authentication. Some applications may not support prompting of the user; in this case you will have to synchronize the passwords between your realm and your Domain.

Neither MIT nor Microsoft provide any tools to perform a password synchronization between a realm and a Domain. Reportedly, CyberSafe does have a tool for their Kerberos implementation that provides this feature.

There are two things that you should keep in mind if you are intending to synchronize passwords between a Domain and a realm. A compromised password in either environment can be used to gain access to resources in the other environment; you will not easily be able to determine where down-level authentication is being used in your environment.

2.5.2 Kerberos referral issues

If you have tried this type configuration yourself, you may have encountered some problems. Just as release to manufacturing occurred I reported to Microsoft that this functionality was not working. It turned out that Microsoft had unknowingly been depending on a bug in the MIT distribution. The bug affected how ticket requests for unknown principals were handled and the implementation of realm referrals. A newer distribution from MIT had fixed the bug and broken the test configuration. The testing had also relied on a hierarchical name relationship between the Domain and realm names, which does not always apply.

Microsoft very quickly proposed a solution that was soon turned into the hotfix described in Microsoft knowledge base article Q253531.

Two configuration options are available with this hotfix.

- If your Kerberos realm does not support name-canonicalization, and the KDC returns a "principal unknown" error in response to a ticket request, the request will be retried to the Domain. This requires the computer issuing the ticket request to belong to a Domain.

- If your realm supports name-canonicalization it must return a referral if the principal is unknown. The client library will then use the referral information for a subsequent ticket request. The machine originating the request must have some registry information present so that the initial request will contain the name-canonicalization bit.

2.5.3 Kerberos name canonicalization issues

The preceding section started off talking about realm referrals but ended up talking about the name-canonicalization. They should be separate topics but their distinction has become somewhat blurred because of the way a bit in the Kerberos options field is being used.

Having the KDC perform name canonicalization is a relatively new concept, introduced by Microsoft. They have submitted a draft to the IETF. The draft has resulted in a lot of discussion, but little closure.

Traditionally Kerberos implementations have performed some name canonicalization on the client. When an unqualified DNS domain name is presented to the client library, the library turns it into a fully qualified domain name. The one problem is that a reverse resolution against DNS is often used and DNS is not secure.

Microsoft has argued that by performing the name canonicalization on the KDC the potential threats posed by DNS attacks are eliminated. This is true. It also true that Microsoft is doing more than their thesis would indicate.

A Microsoft KDC will return some tickets that look very odd to people from the UNIX Kerberos community. For example when requesting service ticket for the machine "foo" Microsoft may respond with a ticket for "FOO$" instead of "FOO.MIT.EDU". This provides Microsoft with some backwards compatibility with protocols that expect NetBIOS style names. It is a neat trick but it does lead to some other problems.

The client libraries are no longer able to perform a simple name matching during their ticket request prior to sending the request. Although a client may already have the needed service ticket, further ticket requests may be issued. Microsoft has acknowledged that this is a minor problem that they discovered late in the development cycle. I have also been told that they expect to address the problem in a future release.

One concern that has been raised by people outside of Microsoft is that using Kerberos as an authentication protocol is well understood, however the implications of using the protocol as a name resolution protocol or directory service are not well understood.

Microsoft has responded by saying that they are not using the protocol as a naming service or directory service. This is true. But I'll bet if you take 10 programmers aside and ask each one how they would solve the cache matching problem describe above, at least of few of them would come up with solutions that sound an awful lot like a naming service or a directory service.

The MIT distribution of Kerberos does not currently include any support for name-canonicalization or the generation of referrals. As mentioned, Microsoft has submitted a draft proposing this extension to the protocol to the IETF but there is no clear consensus within the working group on this proposal yet. As the work proceeds I do expect that the MIT distribution will eventually incorporate the functionality. In the meantime a separate patch will be made available for the MIT distribution so that individual sites may add this support as desired.

2.5.4 MIT library modifications to achieve single sign-on

So far most of our discussion in this section has focused on applications that use the SSPI however, other APIs exist for supporting Kerberos. MIT and other vendors provide Kerberos libraries and GSSAPI libraries that are used by various applications. The libraries and applications work well with the configuration described but the 3rd party libraries and Microsoft SSPI do not share a common ticket cache.

This means that subsequent to the initial Windows 2000 login, the MIT libraries do not have access to the TGT that was obtained by the Microsoft libraries.

Once again Microsoft has listened to the 3rd party development community and helped to provide a solution. Normally only the LSA process has access to the ticket Microsoft ticket cache. This means that hostile applications cannot steal the user's credentials and pass them on to something else. Microsoft has provided an API that enables a user process to copy the TGT from the Microsoft cache and store it in another cache.

MIT is currently working on modifying its Windows libraries to support this functionality. The first time the krb_sendauth function is called the library will copy the initial TGT from the Microsoft cache to the MIT ticket cache. The TGT will then be used to obtain the desired service ticket and complete the krb_sendauth call.

This should result in a single sign-on functionality for many applications. This strategy does not rely on a GINA, which in turn means that sites that rely on other GINAs will not encounter any configuration problem.

I expect this functionality will be available from MIT before the Fall of 2000.

3.0 Microsoft Network Monitor and Kerberos support

Debugging the problems encountered in our configuration and learning about the details involved have been greatly facilitated by the staff at Microsoft and the tools that they provided to us.

We used Microsoft's Network Monitor extensively. The Kerberos version 5 protocol uses ANS.1 encoding which normally makes the understanding the network traffic very difficult. Microsoft has developed a Kerberos parser DLL for their Network Monitor. They graciously allowed us early access to this tool. Their goodwill even extended to letting us redistribute it to a limited number of schools that had existing Kerberos realms and were working on a Windows 2000 deployment. If this parser has not already been released, I expect that it will be within the next several months.

4.0 How do you spell interoperability today?

Ultimately interoperability cannot be declared by a vendor or specified by a standard. The true measure of interoperability can only be performed by each customer or user of a protocol. The question to be asked and answered is, "does this meet my needs?"

This paper only presents some of the possible interoperability scenarios that Microsoft and other Kerberos implementations will be faced with.

If your metric is, "can I choose which vendor provides my KDC?" or "can I provide my users with the functionality of a Windows 2000 domain, without running a Microsoft Domain Controller?", then you are likely to be disappointed by the interoperability provided.

On the other hand, by many other metrics Microsoft has provided a highly interoperable Kerberos implementation that will meet many customers needs.

5.0 References

Windows 2000 Kerberos Interoperability, Microsoft whitepaper <http://www.microsoft.com/WINDOWS2000/library/howitworks/security/kerbint.asp >

Kerberos 5 (krb5 1.0) Interoperability, Technical Walkthrough, Microsoft whitepaper <http://www.microsoft.com/technet/win2000/kerbstep.asp >

Kerberos authentication in Windows NT 5.0 domains, Peter Brundrett, ;login:, the magazine of the USENIX Association, May 1998 Special Issue on Security

J. G. Steiner, B. Clifford Neuman, and J.I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the Winter 1988 Usenix Conference*. February, 1988.

Microsoft knowledge base article Q253531 <http://support.microsoft.com/support/kb/articles/Q253/5/31.ASP >

Enterprise Management of Windows NT Services

J. Nick Otto
notto@parikh.net
Parikh Advanced Systems

Abstract

A problem faced by NT administrators is the management of NT based services in the enterprise. In the NT environment, services have particular startup states and run with varying degrees of security. Multiple NT domains, servers, and workstations providing services all add to the difficulty of managing NT services. Un-authorized introduction of some services can cause problems to end-users and even block network access. Proactive monitoring of NT services is essential to maintaining the availability of network resources.

A system administrator must be aware of the following service related issues: 1) current status and uptime of all critical NT services; 2) services authorized to be running on the network; 3) are there any un-authorized services currently running. A system administrator must perform actions based on facts about network services. However, in order to make quality decisions, information must be collected and monitored, prior to the administrator's involvement. Each area of service management presents a unique challenge. These challenges are the motivating factor behind the tools described in this paper.

1 Introduction

This paper focuses on enterprise-level management of NT services in a large, multi-domain, NT enterprise. The tools and methodologies presented here are applicable in heterogeneous environments. They are designed to assist with the automated management of NT services.

A system leveraging Windows NT server, Visual Basic applications (with Microsoft's Active Directory Service Interfaces (ADSI)), and a Microsoft SQL server back-end was implemented to monitor and manage NT services across the enterprise.

Prior to developing the solution presented in this paper, service management solutions were investigated and tested. And, although there are many inexpensive solutions in the market for managing certain services, a custom solution was the best value. A custom solution allows more control over monitoring and negates post-implementation costs associated with scaling the application, i.e., instance licensing.

2 Systems Requirements

The NT network this solution was built for consisted of approximately 150 NT servers, 500 NT workstations, three domains, and 5,000 clients. The network is distributed over a corporate campus and a Wide Area Network.

2.1 Initial Motivation

Better automation of NT services management became a necessity when a troublesome new service was brought on-line. The service in question would randomly stop running, and require an administrator to start up the service. While working on this issue, it was determined that a tool could be written to check this service each morning and ensure it was running. After developing routines that managed the "problem" service, it was a natural progression to continue development and build a more full-featured service management solution.

2.2 Needs Assessment

The driving forces behind service management issues are NT services that stop for various reasons, fail to start, or are unavailable. Service management is different in each environment; some services are critical, others are not. Frequently services must be stopped for maintenance and should not be re-started until a later date. Based on some of these concepts, conclusions, and ideas, a production system was developed that featured the following:

- A repository of all critical NT services and associated parameters along with customizable option fields
- Administrative front-end, or console, that would allow for easy addition or removal of

monitored services, as well as stopping the monitoring process completely
- An application that would poll services and take action based on repository information
- E-mail notification, configurable per service
- Alpha paging, configurable per service
- Logging of all application functions
- A Web interface to display status, logging summaries, and other reports and statistics
- A solution built completely on Microsoft technology
- A minimal implementation cost

2.3 Requirements Analysis

The main objective of this effort was to develop a system that proactively monitors and takes reactive measures on NT services, resulting in an increase in service availability and a reduction in response times to a service outage.

During the development efforts, prototypes were compiled, posted, and run. Administrator input was gathered from the very beginning of the project and resulted in the addition of many features; most importantly the addition of the error count to limit e-mail and pages.

2.4 Functionality

Proactive service monitoring and the subsequent reactive system responses were a challenge in respect to building the application. Services can be in different states, with varying startup types. An administrator may stop a service for a specific reason and not want it started again. Critical services may require pager notification, but only during certain hours. If a service is off-line and e-mail notification is enabled, the system must be set to stop notification.

The first step in defining functional requirements was addressing the repository. The application needed to operate based on the status of a service as well as the repository information. Each service record in this database contained the following fields: Host Computer, Name, Display Name, Path, Service Account Name, Startup Type, Status, Time/Date, Pager, E-mail, and Error Count.

Services can be in one of seven states; 1) Not Running (Stopped), 2) Start Pending, 3) Stop Pending, 4) Running, 5) Continue Pending, 6) Pause Pending, and 7) Paused. A service is also configured with one of three possible startup states; 1) Automatic, 2) Manual, and 3) Disabled. Using combinations of these various service states, a set of "rules" determined the action taken by the application. The rules function based on a combination of service startup type and service status. If the service has a status of X and startup type of Y, then do Z. The basic rules represented in a case select are shown in Appendix A. Through this set of rules the system takes the appropriate action on a target service. To avoid the system "touching" a service an administrator wants off-line, the startup type can be changed to disabled, a change that can be made quickly and at the same time that the service is stopped.

The system provides for notification of actions taken. Two notifications methods were identified during the needs assessment; e-mail and alpha pager. Sending e-mail is accomplished with a simple SMTP mail routine. Alpha paging is accomplished by tying into a network accessible alpha paging solution. Functionally, the system provides both services, but one would not want to receive hundreds of e-mails if the same service was off-line or to receive excessive pages. An error count field was added to the database to compliment the fields for paging and e-mail. If the error count is above five, e-mail is no longer sent; if above two, alpha pages are no longer sent. An assumption was made that the administrator receiving the pages or e-mails will note the consecutive messages and check the service status.

The database is updated whenever a service is checked. The time and date of last service status check and the last service status are updated. Updating these fields allow for a real-time look at service statuses across the network. Along with updating the main service information, the application logs all actions taken to an error table stored in the database. Error log records contain the following fields: Error Message, Action, and Time/Date. Error log functionality is essential to the system requirements.

An administration program for the main application was developed that enumerates network resources and creates a "tree" view. Using this tool an administrator can check the available services on a server and add, remove, or edit the associated database entries. This tool also allows for "browsing" available resources. Administrators generally know their servers and which machines should be on the network. With a good naming convention one can quickly recognize servers that stand out and then check the services. An added benefit to this functionality is the ability to find rouge network servers, then take appropriate action.

Configuration data for the application is held in the database. Configuration data includes a setting for looping frequency of the service checking routine. On startup the application reads the configuration information, then reads in all services to be checked. Once checked the application "sleeps" for a pre-set amount of time. Each iteration results in a re-reading of configuration information and of services to be checked. With this scenario, an administrator can add or remove services and expect the changes to be applied during the next run of the application.

3 Laying the Groundwork

As defined by the system needs, the service monitoring application was required to be completely built with Microsoft solutions and require minimal implementation costs. Existing development tools and back-end systems were available for this effort and helped meet the requirements. Staying within the Microsoft model allowed the system to be built and implemented on existing servers, with existing Visual Basic licenses, and the database added to an MS-SQL server that served other needs. MS-SQL server 6.5 was the database used for the implementation of this application.

3.1 Why Visual Basic?

Visual Basic 6, Enterprise, was used to develop both the administrator's console and the service monitoring application. Visual Basic provides a rapid development environment for building database driven applications. Visual Basic is easy to learn and is consistent with VBScript allowing for easy transition from application development to Active Server Page Development.

3.2 Microsoft ADSI

The Microsoft Active Directory Service Interfaces (ADSI) are a key part of this application. ADSI version 2.5 was used for this project. The ADSI controls allow objects within the NT enterprise to be connected to and manipulated. With ADSI a developer can bind to a server or the domain and control Services, Users, Printers, Shares, etc. Most core administrative tasks can be accomplished with ADSI and Visual Basic.

Another compelling argument for the use of ADSI for developing administration tools is that ADSI provides support for MS Exchange, Novell NDS, IIS, and LDAP resources. Administration tools built on this technology can bridge systems, unify administration scenarios and reduce the burden of administration in heterogeneous environments.

3.3 The role of IIS and ASP

One of the basic system requirements was to provide a status of services on the network and to provide reports. This application is completely database driven and each error log is written to the database. With all the information in the database it was a logical decision to use IIS with Active Server Pages to provide the interface for checking server status, looking up individual records and generating reports.

4 Implementation

The implementation of the service monitoring application was a straight forward process. Target domains with associated services were identified and documented. Administration groups were notified of the pending installation and trained on what the application would perform. A major function of the application is to start services that are not running. If an administrator is upgrading an application or service and the service monitoring application starts the service, there could be negative effects. Avoiding conflicts between the service monitoring application and administration teams was very important. Thus, the involvement of each administration team was essential to project success.

4.1 System Components

The system consists of four main components: 1) the service monitoring application, 2) the administration interface, 3) the database back-end, and 4) the web interface. The basic system layout is shown in Figure 1.

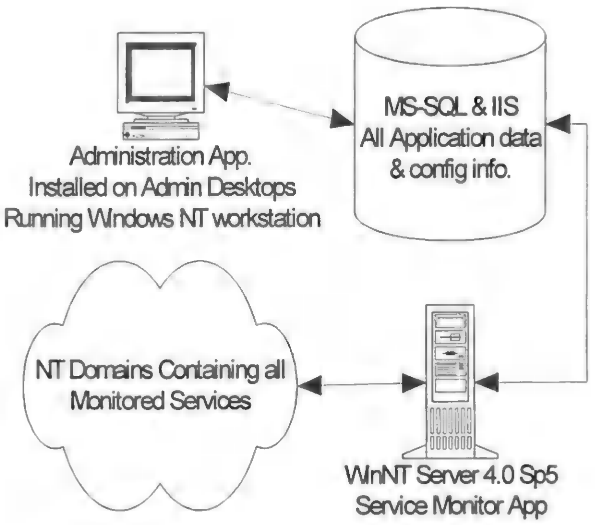


Figure 1 - Basic System Overview

4.2 Security

Since this application was built with Microsoft solutions, security is dependent upon proper settings and configurations in the Windows NT domains where the services managed reside. The services for this implementation resided in three domains, a master with two trusting sub-domains. The service monitoring application is run with domain administrator privileges. Access to the administration program and database are handled via integrated domain security with the MS-SQL server.

4.3 Propagating the database

There were two options available for database propagation: 1) enumerate network resources and add services based on a set of criteria, or 2) selectively add services to the database with the administration utility. Option 2 was chosen for database propagation so that subjective reasoning could be applied to which services were added. Only critical services were added to the database, there was no need to fill the database with extraneous services. If an automatic procedure was used to fill the database, services from all enumerated servers would be in the database. There was no desire to add "test" or other non-critical servers to the program. Also, for each service there is the option for e-mail and/or pager notification upon taking an action on the service. The decision for these settings is completely subjective and up to the administrator.

The process to add all the desired services to the database was approximately a half-day effort. Each server was investigated with the administration program and a decision made on which services should be added and the associated options. After implementation, new services can be added to the database as part of the process of adding a new server to the network.

4.4 Deployment

Deployment of the application took place as the program was being developed. One of the basic system needs was to have a solution that would be installed on currently available hardware. The database was added to an existing server and the application was installed on an administration staff utility server. There was no "roll-out" in the traditional sense. The application did require installation of ADSI on the server that hosted the application. There is no client component to this application. All work is performed by the service monitoring application with all communication and polling originating with the application. This situation is helpful when monitoring services in a firewall DMZ, as all connections originate from inside the network from the server hosting the service monitoring application (although, this was not done for this implementation).

4.5 Testing

Testing consisted of manually changing services to varying states and watching to see if the application behaved as predicted. During this process it was discovered that "hung" services did not always return an expected response to the application, sometimes the service would return nothing. This situation, and the case of a server being off the network, was handled by the error handling routines in the program. If the service didn't return a status, or was not available, a status of "un-reachable" was added to the database and the appropriate notification carried out.

The application proved to be very reliable and accurate in determining what action to take. Many of the monitored services were over a WAN, with some very slow links, and the application handled the slow connections without problems. Sleep routines were built in to account for start-up delays when the application started a service. Originally the application was to issue the start command, then check on the next iteration of the program. However, this makes it more difficult to track what action the program has previously taken. To solve the problem, the program re-checks any service it takes action on, prior to moving to the next service.

4.6 Support Structure and Monitoring

Prior to implementation, service monitoring and management was a reactive process. A service would become un-available and remain so until reported by a user or noticed by an administrator. Service management under the old scenario resided with first level administrators who would typically just re-start the service, (or re-boot the server), and then report the incident to the appropriate group. The service monitoring application changes this scenario in that the re-starting of a service, while still a reactive solution, takes action prior to user notification. This results in minimizing the time a service is un-available.

E-mail and pager notifications are handled by sending the e-mail messages to a mail group that includes all administrators. The "on-call" administrator can check the messages and take action; while other administrators can just delete the messages. This setup avoids the issue of making weekly changes to the e-mail configuration options. It also allows for coverage when the "on-call" administrator is busy and the back-up "on-call" would be seeing the messages. The alpha-pages are sent to an "on-call" pager that is passed around the support group. Pages from the application are treated in the same manner as any other support call.

Monitoring of log files and application activity allows for service monitoring to become a more proactive situation. A given service may stop for an unknown reason on a regular basis. Administrators may be re-starting this service but not readily noticing the trend. Analysis of the log files and system activity allows for trend recognition and association of application activity with other processes. Thus, a problem service may not be the problem at all, just the end result of a different issue. Analysis of log files for this implementation resulted in the identification of several trends that were corrected shortly after implementation.

5 Lessons Learned – Success?

Post-implementation the question was asked, "Were the needs met and does the system work?" The short answer is yes. The suite of Microsoft products chosen proved to be reliable and extensible. Concerns about programmatically managing services were raised and investigated. This simple automation of existing technology proved to be essential for administering large environments where consolidation of effort is important. The application functioned as expected and resulted in identifying troublesome services and root causes. The web interface proved to be a handy addition as this interface can easily be sorted by category. An administrator can check the current status of all instances of a given service, or can just check for any service in the system not currently running.

5.1 Problems and Bugs

The only major problem encountered revolved around network outages. If a WAN link or network segment went down that contains several services, the administration team is hit with many e-mails and pages. This problem is a design issue and can be changed. The work around used for this problem was to stop service monitoring in the event of a network outage. The network in question was very robust and the situation of having to stop service monitoring rarely occurred.

5.2 Extensibility and Future improvements

The main purpose of this application was to monitor and report on services existing within a WinNT domain or domains. However, building the application with ADSI allows for features to be added that can change the application from a service monitoring tool to a service administration tool.

The application can easily be modified to allow for making changes to the service properties, both the administrator's console and web interface can be modified in this way. This would allow an administrator checking service status the ability to change the status with the tool. The startup type and associated account information could then also be modified.

Additionally, ADSI provides support for many other environments and products. Using this support the service monitoring application can be extended to manage resources other than NT services. In a nutshell, the framework chosen for this application allows for extending the application to many aspects of systems administration. Using Visual Basic as the development tool, also allows for taking advantage of the Windows API to add features not supported directly by ADSI.

6 Summary

By leveraging existing resources and the power of Visual Basic with ADSI, a working service-management solution was implemented. Monthly reports show when and where there are issues with key network resources. Troublesome services (those services that tend to stop on their own), can now be re-started expeditiously, reducing downtime and calls to the help desk, and identifying root causes. The logging and tracking functionality provides the ability to quickly recognize trends and problem areas. The implementation resulted in an increased level-of-service and a corresponding decrease in administrative response time.

7 References

15 Seconds Web Page, http://www.15seconds.com.

Eck, Thomas, Windows NT/2000 ADSI Scripting for System Administration, MTP, 2000

Hahn, Steven, ADSI ASP Programmer's Reference, Wrox Press, 1998.

Microsoft Active Directory Services Interfaces Overview pages, http://www.microsoft.com/windows2000/library/howitworks/activedirectory/adsilinks.asp

Appendix A – Code Examples

The code examples show how the decision structure for the service monitoring application works. The GetServiceStatus function is called by the main program to check the current status and startup state of a specified service. If the service is running or paused, the function returns to the main routine. Any state other than paused or running results in the StartService function being called. A rather verbose case select is used during the StartService function to determine if the service starts and if a page should be sent. This code is dependent on several other routines and is not intended to be fully functional.

```
' This function shows the decision structure to decide if the application should take
' an action on a service.  Service to be checked is passed from main routine.
' Return values for service status and start type are defined with constants.
' Actual values can be found in the ADSI documentation.

Function GetServiceStatus(sServiceName As String) As String

Dim ServiceStats As IADsServiceOperations
Dim ServiceInfo As IADsService
Dim Status As Integer

On Error GoTo ErrHandler

Set ServiceStats = GetObject(sServiceName)

Status = ServiceStats.Status

' if the status is running or paused, leave, we're done

If Status = S_Running Then
    GetServiceStatus = "Running"
    iCurrErrCount = "0"
ElseIf Status = S_Paused Then
    GetServiceStatus = "Paused"
    iCurrErrCount = "0"
Else
    Set ServiceInfo = GetObject(sServiceName)
    ' if service is not disabled, pass the service to the startservice function,
    ' if not end routine.
    If Not ServiceInfo.StartType = S_Disabled Then
        GetServiceStatus = StartService(sServiceName)
    Else
        GetServiceStatus = "Stopped(Disabled)"
        iCurrErrCount = "0"
        AddErrLogEntry sServiceName, "Checked service has a start value of Disabled", _
                                      "Re-Start not attempted"
    End If
End If

Set ServiceStats = Nothing
Set ServiceInfo = Nothing

Exit Function

ErrHandler:
    GetServiceStatus = "UnReachable"
    AddErrLogEntry sServiceName, "Service Unavailable or Server off Network", "None"

    iCurrErrCount = iCurrErrCount + 1

    Err.Clear

    Set ServiceStats = Nothing
    Set ServiceInfo = Nothing

End Function
```

```
' this function is used to start services and then pass the results back to the
' getservice status function.  Here again constants are used to identify the return
' value of service status.

Function StartService(sServiceName As String) As String

Dim ServiceOp As IADsServiceOperations
Dim result As String

Set ServiceOp = GetObject(sServiceName)

' We know the service is stopped, so let's start it
ServiceOp.Start

MeSleep (5)   ' wait for service to start

' This case select is used to determine what the application should do
' after the first service startup has been attempted.

Select Case ServiceOp.Status
    Case Is = S_Not_Running
        StartService = "Stopped(Re-Start Failed)"
        AddErrLogEntry sServiceName, "Service not running", "Re-Start attempt failed"
        If iCurrErrCount = "0" And sIsPageable = "Yes" Then
            Send_Page sServiceName & ": Service not running.  Re-Start attempt failed."
        End If
        iCurrErrCount = iCurrErrCount + 1
    Case Is = S_Start_Pending
        MeSleep (5)
        If ServiceOp.Status = S_Running Then
            StartService = "Running(Re-Started)"
            AddErrLogEntry sServiceName, "Service not running", "Re-Started"
            iCurrErrCount = "0"
        Else
            StartService = "Stopped(Re-Start Failed)"
            AddErrLogEntry sServiceName, "Service not running", "Re-Start attempt failed"
            If iCurrErrCount = "0" And sIsPageable = "Yes" Then
                Send_Page sServiceName & ": Service not running.  Re-Start failed."
            End If
            iCurrErrCount = iCurrErrCount + 1
        End If
    Case Is = S_Stop_Pending
        MeSleep (5)
        If ServiceOp.Status = S_Running Then
            StartService = "Running(Re-Started)"
            AddErrLogEntry sServiceName, "Service not running", "Re-Started"
            iCurrErrCount = "0"
        Else
            StartService = "Stopped(Re-Start Failed)"
            AddErrLogEntry sServiceName, "Service not running", "Re-Start attempt failed"
            If iCurrErrCount = "0" And sIsPageable = "Yes" Then
                Send_Page sServiceName & ": Service not running.  Re-Start failed."
            End If
            iCurrErrCount = iCurrErrCount + 1
        End If
    Case Is = S_Running
        StartService = "Running(Re-Started)"
        AddErrLogEntry sServiceName, "Service not running", "Re-Started"
        iCurrErrCount = "0"
    Case Is = S_Continue_Pending
        MeSleep (5)
        If ServiceOp.Status = S_Running Then
            StartService = "Running(Re-Started)"
            AddErrLogEntry sServiceName, "Service not running", "Re-Started"
            iCurrErrCount = "0"
        Else
            StartService = "Stopped(Re-Start Failed)"
            AddErrLogEntry sServiceName, "Service not running", "Re-Start attempt failed"
            If iCurrErrCount = "0" And sIsPageable = "Yes" Then
                Send_Page sServiceName & ": Service not running.  Re-Start failed."
            End If
```

```
                iCurrErrCount = iCurrErrCount + 1
            End If
        Case Is = S_Pause_Pending
            MeSleep (5)
            If ServiceOp.Status = S_Running Then
                StartService = "Running(Re-Started)"
                AddErrLogEntry sServiceName, "Service not running", "Re-Started"
                iCurrErrCount = "0"
            Else
                StartService = "Stopped(Re-Start Failed)"
                AddErrLogEntry sServiceName, "Service not running", "Re-Start attempt failed"
                If iCurrErrCount = "0" And sIsPageable = "Yes" Then
                    Send_Page sServiceName & ": Service not running.  Re-Start failed."
                End If
                iCurrErrCount = iCurrErrCount + 1
            End If
        Case Is = S_Paused
            MeSleep (5)
            If ServiceOp.Status = S_Running Then
                StartService = "Running(Re-Started)"
                AddErrLogEntry sServiceName, "Service not running", "Re-Started"
                iCurrErrCount = "0"
            Else
                StartService = "Stopped(Re-Start Failed)"
                AddErrLogEntry sServiceName, "Service not running", "Re-Start attempt failed"
                If iCurrErrCount = "0" And sIsPageable = "Yes" Then
                    Send_Page sServiceName & ": Service not running.  Re-Start failed."
                End If
                iCurrErrCount = iCurrErrCount + 1
            End If
    End Select

    Set ServiceOp = Nothing

    Exit Function
```

Monitor an Enterprise of SQL Servers - Automating Management by Exception with Perl

Linchi Shea
Merrill Lynch & Co.
linchi_shea@ml.com

Abstract

Monitoring a large number of SQL Servers in an enterprise is a difficult task. The SQL Server administrators have to deal with a large amount of very dynamic and diverse information, and many other complicating factors. We found that the age-old principle of management by exception provides an effective framework in organizing our monitoring efforts. This paper describes our experience in using Perl to help automate the management-by-exception approach to monitoring SQL Servers. Perl is used, in most cases together with SQL, to collect and process SQL Server-related system information, to identify important exceptions, and to report them in a highly summarized fashion. The exception reports and a suite of Perl scripts developed to produce the reports are discussed. We conclude the paper by sharing some lessons learned in our struggle to tame SQL Server monitoring with Perl.

1. Introduction

Monitoring a large number of SQL Servers in a large-scale enterprise environment is a difficult task. When the number of SQL Servers reaches 50+, 100+, and still grows, the nature of monitoring these servers becomes significantly different from monitoring a dozen or two SQL Servers. To begin with, the SQL Server administrator (DBA) must monitor events that are recorded in a multiplicity of sources, and must cope with an overwhelming amount of information. They may have to retrieve relevant system information that does not typically fall in their own realm of responsibilities. In addition, the DBAs are often asked to accommodate a fast growing number of SQL Servers with not-so-fast growing resources.

This paper describes how we use Perl to help deal with monitoring a large number of SQL Servers. The next section identifies several prominent factors that have shaped our overall approach. Section 3 gives a brief summary of a simple three-layered strategy that frames our monitoring efforts. The first layer of the strategy deals with prompt responses to highly critical errors. The second layer provides concise management by exception reporting. The third layer consists of a DBA repository for storing system information from each monitored SQL Server. We use Perl primarily at the second layer to automate the collection, processing, and presentation of SQL Server-related system information for administration purposes. Notice that Perl is not the only language used to enable this strategy. Transact-SQL is also used extensively. But Perl is the focus of this paper. Section 4 covers the Perl scripts used in SQL Server monitoring.

Using this approach, we have been able to accommodate the rapid growth in the number of SQL Servers we have to monitor.

2. Motivation

The last several years have seen a rapid growth in the number of SQL Servers deployed in our environment. It is expected that this number will continue to grow rapidly for the next few years.

In search of a way to monitor these servers, we analyzed potential factors that may influence the effectiveness of any monitoring approach we might eventually adopt. We identified the following to be particularly prominent in our environment:

Information Overload

The amount of information that must be processed to effectively monitor the servers grows much faster than the number of the servers. It is also quite easy to run into the over-monitoring syndrome: monitoring too many things results in too little being effectively monitored.

Information Diversity

Comprehensive monitoring requires system information from diverse sources. They include SQL Server error logs, job log files, job schedules, NT registries, NT event logs, directories, NT service configurations, SQL Server system tables, SQL trace or profiler output, performance monitor counters, SQL Server cluster configurations and state, and so on.

Moving Targets

Monitoring requirements are in a constant state of flux, and the DBAs are in an endless learning mode. The monitoring strategy has to be completely open to accommodate changing requirements, and new experiences and insights.

Distributed Roles and Responsibilities

Though the responsibilities of SQL Server administration is assumed by a dedicated DBA group, it is often not easy to effectively manage SQL Server monitoring-related issues. Communication of these issues can be a challenge. For instance, NT system changes that are managed by separate organizational groups, but may have significant impact on SQL Server, are not always promptly communicated to the DBA group, or not communicated at all. Even within the DBA group, one cannot take communication of SQL Server monitoring for granted.

Enterprise Monitoring Infrastructure

The SQL Server DBAs work in an environment that has a well-equipped, but rapidly evolving, enterprise infrastructure for system monitoring. This infrastructure dictates how critical production errors are monitored. But it is not provided as a solution to meet all the platform-specific or local group-specific monitoring requirements. The SQL Server monitoring strategy must complement this enterprise infrastructure rather than implement a substitute.

Varying Degrees of Urgency

Not all events are of equal criticality. Some events are so urgent that we need to respond right away, even in the wee hours. Responding to other events can wait until the morning, or we simply want to be informed on a daily basis. For events that are neither immediate nor temporarily deferred, we are content with a summary.

If we hope to effectively monitor a large enterprise of SQL Servers and keep up with its rapid growth, we need a strategy to frame our monitoring efforts. In the next section, I outline such a monitoring strategy.

3. Overview of the Monitoring Strategy

In a nutshell, our monitoring strategy is based on the principle of management by exception.

Whatever the strategy may be, the key to its success in this large enterprise environment is automation. We need tools to enable the automation, which adequately accommodate the factors identified above. The tools we use are a combination of a central DBA repository and a collection of SQL and Perl scripts. The repository is made up of (1) a SQL Server 7.0 database, (2) an NT registry hive, and (3) an NT directory tree. The repository stores comprehensive system information for each monitored SQL Server. Perl scripts are used to help extract system information from heterogeneous sources on remote servers, to store the information in the central repository, and to summarize the information by highlighting changes and errors, and identifying patterns.

3.1. Layers of Monitoring

We divide the focus of the monitoring strategy into three layers:

1. Critical production problems
2. Exception reporting
3. DBA Repository

The first layer deals with critical SQL Server production errors that must be addressed promptly. We leverage the existing enterprise monitoring infrastructure to notify the DBAs, and rely on the SQL Server alert mechanism and other enterprise monitoring agents for error detection at this layer. SNMP traps are sent to the enterprise data center, which then notify via pager or telephone the DBAs on an escalation list. SQL Server alerts have been heavily customized to minimize the number of nuisance notifications. The primary objective at this layer is to receive *prompt* notifications on all critical production problems, and on these critical ones only. A full discussion of how this layer of monitoring is implemented is, however, beyond the scope of this paper.

At the second layer, all system exceptions, including the production problems monitored at the first layer, are captured. While at the first layer we'd like to be minimalist in containing the number of notifications, at the second layer we want to be comprehensive in capturing the exceptions that are deemed important. However, we want to report on these exceptions in a highly summarized fashion to reduce information overload on the DBAs.

The third layer of the monitoring strategy is the SQL Server DBA repository itself. At this layer, the objective is to be all-encompassing, i.e. to capture all the system information that may be useful even though it may not be presently used in generating any exception report. This repository also serves other useful purposes beyond reporting on exceptions. For instance, it provides a historical view of system configurations.

The focus of this paper is on the second layer.

3.2. Exceptions

We are interested in the exceptions originated from the following sources:

1. *Operational exceptions.* These are SQL Server-related error conditions. For instance, a table has become corrupted.
2. *Configuration exceptions.* These are significant changes (intended and unintended) in system configurations that affect SQL Server behavior. For instance, an NT administrator has unintentionally changed permissions on the SQL Server data files. Or the SQL Server is changed to no longer listen on an IPC method.
3. *Exceptions to the DBA standards.* These are serious deviations from the established local DBA standards, including standards on configurations, security, best practices, and SQL coding if applicable.

3.3. Exception Reports

The main categories of exception reports that we generate include:

Monitoring System Configurations

We retrieve SQL Server related system information from each monitored SQL Server and store it in the central DBA repository. Retrieved system information includes: (1) the SQL Server registry hives, (2) the SQL Server system tables, (3) generated SQL scripts, and (4) SQL Server-related NT and hardware configurations. We then compare two consecutive daily snapshots of this system information to report any significant changes.

Monitoring Database Schema

Changes in database schema are tracked by comparing two consecutive daily snapshots of the same database schema. Significant differences are highlighted.

Monitoring Security

SQL Server security data is retrieved from each server into the repository on a daily basis. We check the security data in the DBA repository in two areas: (1) we report on any significant changes in SQL Server access and database permissions, and (2) we check and report on the compliance with the established SQL Server security lockdown policies.

Monitoring Performance and Space

Currently, the only performance data collected and reported on is table fragmentation and distribution statistics. Space usage information is extracted from the SQL Server system tables that are already stored in the DBA repository. An exception report is produced when any of the space thresholds is crossed.

Monitoring Logs

In addition to the SQL Servers' own error logs, a large number of log files are generated daily from various scheduled jobs for the DBAs to review. The NT event log may contain additional error messages. All these log files and the NT event logs are scanned to generate a consolidated concise summary.

Monitoring Scheduled Tasks

Depending on the robustness of error trapping, a scheduled job may fail without leaving any trace in its log. The scheduled jobs on all monitored SQL Servers are scanned, and a report on any failed jobs is produced. This also provides a means to monitor the monitoring system since the setup of monitoring uses scheduled jobs heavily.

Monitoring DBA Standards and Best Practices

Not all the DBA standards can be codified in scripts, but a significant portion can be. These scripts highlight whether there is any serious departure from the SQL Server DBA standards or best practices.

4. Perl Scripts

To implement the monitoring strategy of management by exception outlined above, a suite of scripts were written. Most of them were in Perl, some in Transact-SQL, and others in Perl with embedded SQL scripts. Since this paper is about using Perl to facilitate monitoring SQL Servers, I will focus the discussions on the workings of the Perl scripts. All the Perl scripts mentioned in the paper are listed in the appendix. Some sample reports from these scripts are also shown in the appendix.

4.1. DBA Repository

The DBA repository is made up of three collections of information on a dedicated SQL Server: (1*) a SQL Server 7.0 database*, (2) *a registry hive*, and (3*) a tree of NT directories*.

The SQL Server 7.0 database contains tables mirroring all the significant SQL Server system tables. The data is updated daily and kept for 30 days (configurable). Examples of the system tables include sysprocesses, syslocks, sysdatabases, sysdevices, and syslogins at the server level as well as sysobjects, sysprotects, and sysindexes at the database level. In addition, several tables record NT level information such as space on each drive, CPU count, and memory, as well as summary information such as database space consumption.

The registry hive contains the last five versions of SQL Server registry keys and values. The following registry hives from each monitored SQL Server are copied to the registry of the DBA repository server[1]:

- HKLM\Software\Microsoft\MSSQLServer,
- HKLM\System\CurrentControlSet\Services\MSSQLServer,
- HKLM\System\CurrentControlSet\Services\SQLExecutive

They are copied under the key

HKLM\Software\SQLServerAdmin\Monitored Servers\<Server Name>\<Date String>

Registry is used as a part of the DBA repository because some important SQL Server configurations are stored in the registry. There is no point of converting these SQL Server registry keys and values into any other storage format.

The NT directories contain generated SQL scripts for re-creating database devices, backup devices, server configurations, database schemas, and SQL Server scheduled tasks. The directory tree also contains exported system tables in plain text files. In addition, a SQL Server setup initialization file is generated daily. This is mainly for convenience because the same information is already captured in the registry hive and the system tables. Furthermore, complete database access information is dumped out daily to a text file. These files are kept for 30 days.

The system table information is conveniently retrieved into the DBA repository using SQL Server remote stored procedure calls or linked servers. The generated SQL scripts and the exported system tables are produced by the GenerateSQLScripts.pl and ExportSystemTables.pl scripts. The SQL Server setup initialization file is generated by script GenerateSQLSetupIniFile.pl. File extension pl indicates that it is a Perl script.

4.2. System Configuration Changes

Every day after the SQL Server registry keys and values are copied, script CompareSQLRegKey.pl is run to compare the most recent two versions of the registry keys and values for each monitored server. A summary report is produced if there is any significant change.

The CompareSQLConfig.pl script is run daily to identify significant differences between two consecutive copies of the generated SQL scripts for setting the server and database configurations. This helps identify, for instance, whether the SQL Server security mode is changed, a new data device is added, a new task is scheduled, a database option is changed, or even a new database is created.

4.3. Database Schema Changes

There is often a need to compare the schemas of two databases to highlight any differences or to compare the schemas of the same database over time to identify any unauthorized changes. There are many existing approaches with varying degrees of success. Most of these approaches compare system tables through SQL queries. We found that, with the help of Perl it is much easier and versatile

[1] Most examples in this paper assume that SQL Server 6.5 is monitored.

to compare two copies of properly generated database schema scripts.

The CompareDBSchema.pl script executes in two steps. In the first step, it talks to SQL Server through the SQL Server distributed management objects (SQLDMO) to generate SQL scripts for the database objects. We control how SQL scripts are generated for which objects, depending on what we want to compare. The second step does a quick parse of the scripts and stores the result in a Perl nested hash record structure. For our purposes, there is no need to write a sophisticated parser to parse these SQL scripts. Because we control how they are produced by SQLDMO in a predictable format, Perl can parse them very conveniently. The Perl script compares the two hash record structures according to some comparison rules. These rules can be easily added to or removed from the Perl script.

Notice that for planned schema changes, this would provide additional validation that the changes have indeed taken place. At the risk of stating the obvious, the facility is not meant as a substitute for rigorous change control management.

4.4. Performance and Space Problems

Since the information on both database and disk space for each monitored SQL Server is already collected in the DBA repository, producing exception reports when the space consumption has crossed a threshold is done easily with SQL scripts alone.

To check table fragmentation, we run script AlertTableFrag.pl that executes an embedded SQL script in each user database. The SQL script produces, in a temporary file, the result of checking fragmentation against every user table. The Perl script then scans through the temporary file, reporting on large tables with fragmentation (i.e. scan density ratio) that exceeds a threshold.

Similarly, script AlertStaleStats.pl runs SQL Server distribution statistics utility against every user table index. It then scans the result for large table indexes whose statistics have not been updated for a period of time that exceeds a threshold.

We are also working on a Perl script that would automate the processing of SQL Profiler/Trace files and highlight significant performance variances for the same queries or stored procedures.

4.5. Log Files and NT EventLog

In our environment, the DBAs must review a large number of log files each day to identify problems or verify that the systems are in good health. If done manually, this process is boring and laborious, and may result in the log files not being effectively reviewed at all. We need a way to condense this vast array of log files while making sure that no significant errors would be overlooked. This is achieved with script CheckSQLErr.pl, which processes the log files as follows:

- The script summarizes the SQL Server startup process by identifying the SQL Server startup date/time, and reports any failure to listen on any configured IPC method.
- For each SQL Server error number, the script reports the total number of occurrences of the error and the last time it occurred as well as the error message text from the latest occurrence.
- For deadlock error messages, the script reports the total number of times deadlocks have been detected by SQL Server as well as the last time a deadlock occurred.
- The script takes advantage of the fact that every database backup is recorded in the errorlog. If a database has not been backed up within the last 24 hours (configurable), this information is noted in the summary.
- Other messages/errors (e.g. KILL messages, DB-Lib problems, and ODBC connection errors) are treated similarly. This list is easily extensible by adding new regular expression patterns, whenever we see fit.

Words/lines that are not needed in the summary are skipped, and the rest are stored into a Perl nested hash record structure. The script then prints the summary report from the record structure. The script can be easily modified to decide what should or should not be skipped or printed.

The NT EventLog is summarized similarly for SQL Server related errors and warnings using CheckNTEventlog.pl. SQL Server error messages that are written to both the SQL Server error log and the NT EventLog, and therefore already reported by CheckSQLErr.pl, are ignored.

4.6. Security and Access Exceptions

Script CheckSQLLockdown.pl checks each server for significant deviations from the established DBA security policies. For instance, it checks for the following (not an exhaustive list):

- Presence of SQL Server logins with null password
- Guest account
- Whether any SQL Server user can automatically get local NT administrator 's privileges through shell escape (i.e. xp_cmdshell)
- A list of NT accounts with sa access to the SQL Server through NT authentication
- Remote servers with sa access to this SQL Server via remote stored procedure calls

A second script, CheckSQLSecurityChanges.pl, reports on significant changes of security setup in the above listed areas, and database access changes in general. The report is produced again by comparing two consecutive copies of security setup and access information output by a SQL script.

4.7. Exceptions to DBA Standards and Best Practices

When we establish DBA standards and best practices, we intend to have them followed, or broken for good reasons. If they are broken, we definitely want to be informed. The CheckSQLStdDev.pl script scans SQL Server to report on deviations from the DBA standards and best practices.

The following is a sample list of DBA standards and best practices that are codified in script CheckDBAStdDev.pl:

- *Hardware and NT configurations.* Is the time on SQL Servers synchronized? Are the data files placed on a uniform level within the directory tree? Is compress turned on for the drive that stores the database files? Are consistent file extensions used?
- *Server- and database-level configurations.* Is the amount of memory allocated to SQL Server within the reasonable range? Is the master device marked default? Are data files and log files for a database placed on separate disks? Is a disk device shared by databases? Do data and log share the same device for a database? Is there a device not used by any database? Is auto shrink turned on? Are there too many data files for a database? Is a database set to grow in small increments?
- *Database schemas.* Is there a new user object created in the master database? Are there any users with mismatched logins? Is there any index with very low selectivity? Are there any orphaned users? Are there any duplicate indexes? Are there any cluster-indexed columns that are updated frequently? Is there any very wide clustered index with multiple non-clustered indexes on the same table? Is there any procedure/trigger using 'SELECT * FROM' or doing INSERT INTO <table> without identifying column names?

It is worth pointing out that what we consider should be on the list of the DBA standards and best practices are constantly changing.

5. Lessons Learned

Our experience so far indicates that organizing Perl scripts under an articulated framework to accomplish specific SQL Server administration tasks is a powerful concept. We have been able to scale this approach in three directions. First, we started with monitoring a relatively small number of SQL Servers. Adding more servers results in little extra monitoring effort. Secondly, the SQL Server DBA group has grown in size. These exception reports prove to be an effective communication tool within the group. Thirdly, with the repository containing comprehensive system information, we have been able to create new exception reports easily to cover the areas we overlooked.

Perl is integral to the success of this strategy. The large number of modules that come with the ActiveState distribution or available from CPAN makes it easy to extract system information from such diverse sources as NT registries, NT event logs, SQL Server databases, and text files, and to obtain information on hard drives and filesystems. It is very convenient to wrap Perl around NT utilities such as *sc.exe* (a versatile NT service configuration and query tool from the NT resource kit) to get system information. Similarly, we take advantage of the text processing power in Perl to combine the results from multiple SQL scripts into a desirable presentation.

For the kind of monitoring discussed in this paper, we found that it is often simpler to first run a SQL script through *isql.exe*, and then use Perl to process the well-formatted results. In many cases, this is much easier than going through a row-oriented database access module like Win32::ODBC.

Why not use third party tools? Using third party tools to execute the same tasks accomplished by the Perl scripts described in this paper would require an expensive potpourri of tools. We are not aware of any single commercial package that does this type of monitoring comprehensively, or that can be easily tailored to our specific requirements.

Why not use a scripting language like VB? The intent of this paper is to demonstrate the value of Perl in helping automate the SQL Server monitoring tasks rather than discount the value of other languages. It would be interesting to see how any other commonly used language or tools come close to the power of Perl in text processing and pattern matching, and the ease of Perl to glue together the results of wide-ranging tools.

Perl is not the most convenient tool for everything. This is obvious, yet easily overlooked when one is totally engrossed in Perl. In our case, for instance, some exception reports are much more easily produced with SQL queries alone.

Very gladly, we found that many of these Perl scripts are useful beyond monitoring SQL Servers. For instance, it is common for us to receive ownership of a SQL Server installation that has been running without administrative care. We are able to run some of our system monitoring and security monitoring scripts to give us a quick inventory of the system and to identify areas we should pay attention to bring the system in line with our standards.

As a bonus, we also found that our monitoring strategy enables us to maintain what we call 'live documentation'. Since all the system information is in the repository, we can run a script against it to generate up-to-date documentation for any server we are monitoring.

Constantly, we are being reminded that there is no such thing as 'the production release' of the monitoring solution. We are forced to regularly modify our scripts to incorporate new experiences, to adapt to changes, to cover new requirements, and to correct false assumptions, in particular, assumptions on what is worth noting and what is simply a nuisance. The combination of Perl and SQL proves to be extremely convenient in this regard.

Finally, it should be stressed that the approach described in this paper should be applied only to monitor the production environments where changes are introduced in an orderly fashion and any uncontrolled changes are truly exceptions. Otherwise, the exception reporting can be overwhelming and defeats its very purpose.

6. Conclusions

Writing Perl scripts to automate large-scale system administration is not new at all. People have been doing this since the inception of Perl. Unfortunately, though, this still seems to be a rather foreign concept to most Microsoft SQL Server DBAs. Hopefully, this discussion of automating SQL Server monitoring with Perl has demonstrated the power of Perl in improving SQL Server administration.

7. Appendix

7.1. Perl Scripts

The following is a list of Perl scripts mentioned in this paper. They can be obtained from the author:

- GenerateSQLScripts.pl
- GenerateSQLSetupIniFile.pl
- ExportSystemTables.pl
- CompareSQLRegKey.pl
- CompareSQLConfig.pl
- CompareDBSchema.pl
- AlertTableFrag.pl
- AlertStaleStats.pl
- CheckSQLErr.pl
- CheckNTEventlog.pl
- CheckSQLLockdown.pl
- CheckSecurityChanges.pl
- CheckSQLStdDev.pl

7.2. Perl Modules

The following standard Perl modules are used in the scripts mentioned in this paper:

- Win32::Registry
- Win32::EventLog
- Win32::OLE
- Win32::Service

The following modules are also used, and are available from www.roth.net:

- Win32::AdminMisc
- Win32::Perms

In addition, I collected common Perl routines used in performing DBA work into the following module:

- SQLAdmin::DBA

All Perl scripts listed above in Section 7.1 use this module.

7.3. Sample Exception Reports

To illustrate the exception reports produced by the Perl scripts, we give some sample here. Notice that these reports have been reformatted and abridged to fit the required layout of this paper. Each sample report is followed by a brief explanation, and boxes are used to draw attention to certain salient items in the reports.

7.3.1. CompareSQLRegKey.pl

```
***Alert SQL Registry Changes

[ABCSQL01]
[Diff btw 2000-04-12 & 2000-04-13] =>{
  [Software] => {
     [Microsoft] => {
         [MSSQLServer] => {
              [Client] => {
                [ConnectTo] => {
                  <>value: ABCSQL02
                }
              }
              [MSSQLServer] => {
                [Parameters] => {
                  + value: SQLArg2
                }
              }
         }
     }
  }
}
```

This report shows that between April 12, 2000 and April 13, 2000 on SQL Server ABCSQL01, the client network configuration to server ABCSQL02 was changed and a new startup parameter (SQLArg2) was added.

7.3.2. CompareDBSchema.pl

```
***Alert SQL DB Schema Changes ...

[ABSQL01.CaseDB]
Diff btw 2000-04-12 and 2000-04-13

***Msg: DB objects in 2000-04-12 but
   not in 2000-04-13:

 Procedure      dbo.sp_TableLoad
 Procedure      dbo.sp_dbcc
 Table          dbo.AppConflict

Total # of DB objects in 2000-04-12
but not in 2000-04-13 = 3

Total # of DB objects in 2000-04-12
but not in 2000-04-13 = 0

***Msg: Scripts for these common DB
   objects are different:

 Procedure      sp_CheckSum
 Procedure      sp_BatchUpdate
 Table          tb_Employee
```

This report highlights the differences in database CaseDB schema between April 12, 2000 and April 13, 2000. For instance, procedures sp_TableLoad, sp_dbcc, and table dbo.AppConflict were dropped between April 12, 2000 and April 13, 2000. It also shows that no new objects were added. Moreover, it indicates that even though procedure sp_CheckSum and sp_BatchUpdate, and table tb_Employee are in the database on both April 12, 2000 and April 13, 2000, they are modified.

7.3.3. AlertTableFrag.pl

```
***Alert Table Fragmentation

[ABCSQL01.CaseDB]
  dbo.Accounts                 80.00%
  dbo.TranMaster               77.00%
```

This report shows that two tables, Accounts and TranMaster, in database CaseDB on server ABCSQL01 are considered large (e.g. > 100,000 rows) and are significantly fragmented (the main indicator scan density ratio < 85%).

7.3.4. CheckSQLErr.pl

```
[ABCSQL01]
Chking Errorlog D:\MSSQL\LOG\Errorlog
Restarted at: 2000/02/20 14:55:45
```

```
Err: 1608, Severity: 21, Total #: 36,
   Last Occurred: 2000/04/26 11:47:06,
   A network error was encountered
   while sending results to the front
   end.

Chking DBCC log D:\DBA\LOG\DBCC.LOG,
   Created 0 days ago
CaseDB, Msg: 2540, Severity: 16, Total
   #: 1, Allocation Discrepancy: Page
   is allocated but not linked;
tempdb, Msg: 2540, Severity: 16, Total
   #: 79, Alloc Discrepancy: Page is
   allocated but not linked;
tempdb, Msg: 2546, Severity: 16, Total
   #: 1, Table Corrupt: Extent id 256
   on alloc pg# 256 has objid -641,
   used bit on, but reference bit off.

[ABCSQL02]
Chking Errorlog D:\MSSQL\LOG\Errorlog
Restarted at: 00/02/10 08:36:35

Chking DBCC log E:\DBA\Log\DBCC.LOG,
Created 0 days ago

DB: tempdb, Msg: 2540, everity: 16,
   Total #: 1, Allocation Discrepancy:
   Page is allocated but not linked;
Warning:  dump tran with no_log issued
   by 'sa' invalidates log dumps in
   database 'Loans' taken after Apr 27
   2000 6:05PM, Total #: 1, Last
   Occurred: 2000/04/27 18:15:11
***Deadlock detected, Total #: 46,
   last occurred: 2000/04/30 17:24:34
```

Among other things, this report summarizes the following: SQL Server ABCSQL01 is last started on 2000/02/20, and so far, it has encountered 36 errors of error number 1608. It highlights when a log file is created. It also shows that the SQL Server has encountered a total of 46 deadlocks and the last one took place at 17:24:34 on 2000/04/30.

7.3.5. CheckSQLStdDev.pl

```
[ABCSQL02]
***Warning: 16MB out of 512MB
   allocated to SQL Server

***Warning: These data files are in
   the root directory:
       D:\data1.dat
       D:\data2.dat

***Warning: The following devices are
   not used by any database:
TestCaseDB_Dat01,
   D:\MSSQL\Data\TestCaseDB_Dat01.DAT
TestCaseDB_Log01,
   E:\MSSQL\Data\TestCaseDB_Log01.DAT

***Warning: Master device is default.

*** Warning: these devices are shared
   by different databases:
   DeviceName       DatabaseName
   --------------   -------------
   DBA_DAT01        DBA
   DBA_DAT01        ProjectDB
   DBA_LOG01        DBA
   DBA_LOG01        ProjectDB
   master           master
   master           model
   master           pubs
```

The report identifies the potential violations of the following DBA best practices:

- Giving 16 MB out of 512 MB of available RAM to SQL Server is unusually low.
- We do not advise placing data files in the root directory.
- If a device is not used by any database, it should be removed. Otherwise, it's just a waste of space.
- The master device should not be the default data device.
- No device should be shared by different user databases.

Automated Generic Operating System Installation and Maintenance

Joel D. Martin, jmartin@css.tayloru.edu
Compaq Computer Corporation
Aaron D. Brooks, abrooks@css.tayloru.edu
Taylor University CSS Department

Abstract

Microsoft Windows NT deployment and maintenance is one of the most time consuming tasks for systems administrators. The primary motivation for the JACAL project is to streamline this process by creating a set of free and open tools and guidelines for automating the installation and maintenance of one or more operating systems in large computing environments.[1]

It is well known that Windows NT workstations have subtle and potentially serious problems when they are duplicated from a disk image. These problems leave IT administrators with the task of manually setting up NT workstations or cobbling together a number of different automation methods which, in the end, still require the administrators to manually fill in the parts which the various automation tools leave out. Most large scale installation and maintenance tools do not provide solutions that solve the problem completely from boot to a completed build with minimal administrator intervention.

The JACAL project was developed at Taylor University's Computing and System Sciences Department. Our lab and classroom environment consists of 40 workstations that dual-boot Windows NT and Linux. We also have 5 dedicated Linux workstations, 12 dedicated NT workstations, and 7 faculty workstations. Each Windows NT setup has nearly 80 applications that are available to every user. Nearly 1 GB of application data is installed to the local hard drive and over 7 GB of application data is served to the workstations from the application server. Manual installation of these applications takes 25+ hours on a single machine. JACAL allows us to fully rebuild a workstation in under 2 hours with only a few minutes of administrator intervention.

1 As we have developed this paper the emphasis has gradually shifted. Most of the time spent in developing JACAL was spent getting around architectural and common practice problems with Windows NT. Likewise, this paper focuses on the work that we did on the Windows NT side of the JACAL project. A more appropriate title for this paper might be "Applying Linux/UNIX Tools and Methodologies to Aid in Windows NT Installation and Maintenance."

Requirements

Although our environment is not especially large in terms of the number of seats, it is very complex in terms of the number and variety of applications available from each workstation. Our requirements for the JACAL project include the following:

- JACAL should provide a quick and flexible way to rebuild all of our workstations with minimal administrator intervention. This system should be applicable to other environments.
- The tools used should be freely available so that JACAL can be used in organizations with small budgets.
- The tools used should be Open Source where possible to allow for third party customization and extension.
- JACAL should allow Windows NT applications to be installed so that most components of the application can be run from the application server.
- JACAL should provide a simple way to add and repair applications on the workstation.
- JACAL should support the installation of other operating systems.
- JACAL should be able to install multiple operating systems on the same workstation (multi-boot)
- Application images created with the JACAL system should not be order dependent. They should be installable in any order.
- JACAL should resolve the problem known as "DLL Hell". In other words, JACAL should provide a method for resolving shared file conflicts.

The Problem

According to Microsoft, Windows NT 4.0 should not be duplicated from a disk image because of the resulting security issues [SID]. While this is true, most of the security issues can be cleaned up by a SID (System ID) changer such as the one available from sysinternals.com [SYS]. The real problem with "ghosting" an NT workstation is that the NT setup process ties itself very tightly to the particular hardware configuration that it is installed on. This means that a "disk image" must be created for each hardware configuration on the network.

JACAL uses NT's built–in unattended setup process which uses an answer file[2] to automate the process [BELL]. The unattended solution is less than ideal because the process is much slower and less flexible than being able to perform the necessary steps directly from a JACAL script. However, this appears to be the only acceptable solution and the future does not look much brighter. The tools included with Windows 2000 appear to promote the same unattended methodology included in NT 4.0 [UNA1].

The lack of flexibility of disk duplication[3] is not acceptable for our diverse environment because we depend on the ability to swap similar but different components between our many different workstations. JACAL allows us to rebuild our NT workstations very quickly on new hardware configurations without sacrificing flexibility.

Even more serious than the problems with Windows NT setup is the installation and maintenance of NT applications. Gomberg, Evard, and Stacey in their paper [ANL] outline many of the problems related to large scale application installations on Windows NT. These problems include:

- NT applications are more complex and monolithic than typical UNIX applications.
- NT applications are tightly integrated with the OS.
- NT software is oriented to a single–machine and single–user environment (non–networked).
- NT installation programs are GUI based making software installation more difficult to extend and automate.
- Each software installation tunes itself to the particular NT setup, registry and hardware configuration.

DLL Hell

One of the largest problems that JACAL solves is what Microsoft refers to as "DLL Hell"[HELL]. The problem stems from the monolithic nature of Windows applications and the tight integration between the applications and the OS. New applications tend to carry shared file "baggage"[4] that can easily replace shared system files. This can potentially break other applications that depend on certain versions of those shared files.

2 unattend.txt

3 Tools that accomplish this include Ghost and ImageCast

4 This baggage consists of shared files that the software company did not create but included with the application installation. Typically these shared files originate at Microsoft.

UNIX systems also have administrative issues in relation to shared files. However, typical UNIX applications do not actually carry this shared file baggage and therefore do not have the potential to break other applications. Instead of breaking other applications, the new application will simply not work with the current shared files. These problems can usually be resolved with symbolic links.

The Options

Microsoft's System Management Server

The problem of large automated installs has a long and rich history. Microsoft has many pages of online documentation relating to the automation of large NT installations. Microsoft suggests using System Management Server (SMS) which provides remote administration of NT machines including installation of software. We do not consider SMS a viable option for several reasons. SMS is prohibitively expensive for our environment [SMS]. SMS does not provide the ability to build NT (or other operating systems) from scratch. SMS and the other solutions that we looked at do not perform the shared file conflict resolution that JACAL performs. In addition, JACAL now has the ability to do large–scale remote maintenance of NT workstations with a combination of our "wintel.pl" script and the NT Telnet Server "NDTelnet" [NDT]. This is described in more depth in the Detailed System Description.

Bell Lab's AutoInstall for NT

In a paper by Fulmer and Levine of Bell Labs[BELL], they describe a system that is somewhat similar to our JACAL project. This paper was written in 1998 and their system has probably developed considerably since that time. There are many differences between JACAL and the system that they described. The bootdisk system at Bell labs used a MS–DOS bootdisk to initiate the setup process. One of the challenges that they faced was getting all the necessary tools to fit on a single floppy. We have avoided this problem by having only a Linux kernel on our bootdisk which mounts its root filesystem via NFS. Because of this our tools and scripts are not limited by the size of a floppy disk. We are also able to use the JACAL bootdisk to support the installation of other operating systems.

UBS AG's State Driven Solution

One of the more promising solutions was a state driven system recently documented by Martin Sjolin [STATE]. This state driven system stores application configuration information in a central repository which is queried by a client side component whenever an update or installation is performed. This state driven system corresponds to the second phase of our JACAL system. It assumes that the client workstations have a complete NT setup. Their state driven solution quite possibly more flexible than our system and provides more administrative maintenance tools. However, it is unclear whether shared file conflict resolution could be easily accomplished using this system. In the future we may incorporate some of the state driven features of the system into the JACAL project.

The Solution

To meet our requirements for JACAL, we developed an internal system at Taylor University's Computing and System Sciences Department which allows us to quickly rebuild, from scratch, every one of our lab and classroom machines with both Linux and Windows NT (dual–boot) and a full set of applications on both.

Many of the past solutions focus on using the installation tools that are preferred for each application such as InstallShield, a custom setup program for that application or the new Windows Installer. However, in our environment, this is unacceptable because we need more control over the configuration of the target machines. We have approximately 80 applications installed on each workstation. Without our conflict resolution system, the sheer volume of applications installed resulted in constant conflicts as new applications replaced shared files.

We have developed a system based on Microsoft's "sysdiff" program[5] that adds the capability to resolve shared file conflicts. The addition of this functionality has made the program suitable for nearly all of our applications. This conflict resolution capability is one of the biggest differences between our solution and other solutions that have been implemented in the past.

NT Service Packs[6] are the main exception to our conflict resolution system. Microsoft service releases make changes that are too dynamic and far reaching to be properly replicated by a system difference tool such as sysdiff.

JACAL Overview

There are two phases to the JACAL system: the JACAL loader phase and the individual OS and applications install phase. We currently implement the **first** phase as a single bootable floppy with a Linux kernel that NFS[7] mounts its filesystem. The JACAL loader scripts are then executed. At this point the administrator is asked for the machine name, and is given a choice of several different standard machine setups. The options define partition sizes, applications to install, and whether to make the machine multi–boot or dedicated, or to simply refresh the existing machine setup.

Setting up the local Linux partition(s) is a simple matter of using Andrew Tridgell's popular rsync tool [RSYNC] to mirror our existing Linux filesystem. The Linux configuration is then localized completing the Linux installation. The loader phase also copies the Windows NT install files to the workstation in preparation for the second phase.

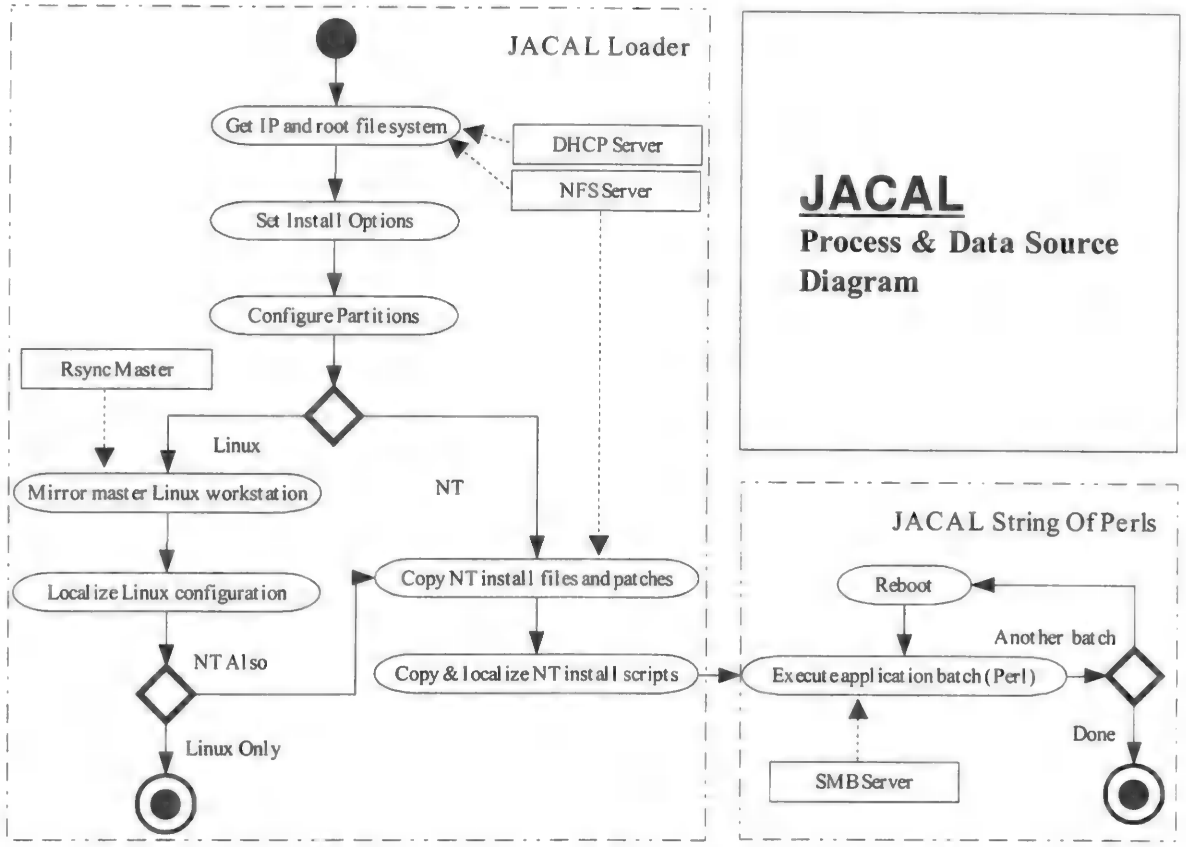
Since NT cannot be fully setup from the loader phase, the **second** phase is specifically related to Windows NT. This phase could easily be used to install any other OS that has "ghosting" difficulties and has the ability to perform a scripted installation. The second phase begins when the loader phase reboots the machine. The Windows NT unattended setup is executed at this point.

When Windows NT has completed its setup, a string of Perls scripts are executed that install applications, Service Packs, and drivers that cannot automatically be specified in the Windows NT unattended setup answer file.

The unattended installation capability of Windows and its service packs are built into the software. We do not install regular applications in this "unattended" method. Instead, we use Microsoft's sysdiff utility to capture the changes that an application makes to a bare system. We then do some semi–automated processing of this "diff" output. First, we change the registry and Windows shortcut paths to allow the application to run from a network server. Then we process the captured directory structure and resolve shared file conflicts, such as DLLs, using a script that allows us to symlink[8] these files to the chosen version of the file. When the application is installed the symlinks are dereferenced and the chosen shared file is used.

At the end of the installation process the administrator is only required to change the local administrative password to complete the workstation setup. JACAL also provides post–setup maintenance tools[9].

5 This tool is now called "Discover" under Windows 2000
6 We consider Microsoft Internet Explorer to be equivalent to a Windows Service Pack
7 Network File System common on UN*X networks
8 UN*X platforms allow pseudo–files, called symlinks, which point to the original file. This referencing is transparent unlike Microsoft ".lnk" files
9 These tools include the Perl based scripts "wintel.pl" and "reghack.pl"


JACAL Loader
Get IP and root filesystem
DHCP Server
NFS Server
Set Install Options
Configure Partitions
Rsync Master
Linux
NT
Mirror master Linux workstation
Localize Linux configuration
NT Also
Linux Only
Copy NT install files and patches
Copy & localize NT install scripts
JACAL
Process & Data Source Diagram
JACAL String Of Perls
Reboot
Another batch
Execute application batch (Perl)
Done
SMB Server

Detailed System Description

(See above diagram)

JACAL Loader

1. (25 Seconds) Boots Linux kernel off of floppy disk
- The 3½" floppy is ext2fs formatted with only a Linux kernel booted by lilo
- The kernel is compiled with the following
  - NFS root
  - NFS client
  - Kernel auto-configuration
  - DHCP/Bootp[10] client
  - As many NIC[11] cards compiled in as the administrator would like[12]
  - This disk is all that is needed to start the setup on the client end
2. (20 Seconds) Runs Linux startup environment
  - Gets IP address from DHCP
  - Mounts linuxroot export from NFS Server
  - Runs modified init scripts
3. (0-5 Seconds) Automatically detects or queries administrator for machine identity

10 Dynamic Host Configuration Protocol/Boot Protocol
11 Network Interface Card
12 In our case, all ethernet cards

- Can be fully automated
  - Database lookup of MAC ID from network card or similar hardware ID (CPUID)
  - If DHCP is setup to assign the same address to the same machine, DNS[13] can be used to figure out machine name
- Administrator can be manually queried for
  - Machine name
  - Configuration (select single menu item)
- Target drive is partitioned if need be
  - Partitioning can be relative based on hard drive size
  - Swap partitions can be created dynamically based on the amount of RAM
4. (25 Minutes *optional*) Linux Install[14]
- Format ext2fs partition (if needed)
- Rsync filesystem from server
- Modify configuration files to reflect machine identity
5. (5 Minutes *optional*) NT Partition Preparation
- Format FAT16 partition (Convert to NTFS later if desired.)
- Copy NT installation files

13 Domain Name System
14 25 minutes is based on a "from scratch" Linux installation. If the partition already has linux rsync will often take less than 5 minutes

- Copy NT installation patches
- Parse and customize NT installation scripts and configuration files
- Run spp client to reset domain machine account password on the SaMBa server

JACAL String Of Perls (Windows NT Installation)

In order to run this step the JACAL Loader (above) must have prepared the NT target partition. Quite possibly the most frustrating problem for an administrator is how often Windows NT 4.0 requires a reboot of the system. Events like changing the cache directory for Internet Explorer 5 to another directory requires a subsequent shutdown and reboot.

When automating the NT installation process, it became apparent that the first thing we would have to [conquer] would be constant administrator intervention to start the next phase of the install after a reboot.

The "String of Perls" in this phase of the installation refers to a customizable reboot management system built on RunOnce[15] registry keys, batch scripts and Perl scripts. (Initially there were only Perl scripts, now most of the .pl's are .cmd's which are faster, lighter and still do the job.) These scripts insert the RunOnce key for the following section of the install and then run the next step in the installation process which requires a reboot of the machine upon completion.

The NT SOP install takes 45 minutes on a single machine.
1. (5 Minutes) NT unattended install
2. (5 Minutes) NT Service Pack 6
3. (5 Minutes) Internet Explorer 5
4. (25 Minutes) Application installations
5. (2 Minutes) NT SMP kernel installation
6. (2 Minutes) Display adapter installation
7. (1 Minute) Cleanup scripts

The last step is only included here for accuracy. At the time of this writing the cleanup scripts run after a reboot which is entirely unnecessary. This step can be merged with the preceding step. Cleanup includes removing installation files and directories from the target drive and changing the password for the "Administrator" account. At the current time the administrator password must be set manually. This is the only human intervention required after the initial 45 second floppy disk boot.

The total sum of human interaction is listed below:
1. Place floppy disk in drive and power up machine
2. Remove floppy disk after floppy access completes
3. If install process is not set up to automatically detect machine identity, type in machine name and select machine configuration from numbered menu.
4. Play Quake Arena[16] for 25 to 75 minutes depending on install[17]
5. If NT installation, change administrator password

Application Image Creation

The process used to create an application image is described in detail on the JACAL website. The basic steps that we use to create NT application install images and where the steps are performed are as follows:
1. (NT) Snapshot a clean system using "sysdiff.exe /snap"
2. (NT) Install the application locally
3. (NT) Capture changes made by the install using "sysdiff.exe /diff"
4. (NT) Extract the "difference" files to the SMB install share using "sysdiff.exe /inf /m"[18]
5. (NT) Rename the "difference" files to use long filenames using "parse–rename.pl"
6. (NT) Repath Windows shortcuts (*.lnk) using "repath–shortcuts.pl"
7. (UNIX) Repath the registry settings file (.inf) and Windows .ini files using regular expressions
8. (UNIX) Create the SMB execution share and move the main application directory to the share
9. (UNIX) Resolve shared file conflicts using "mvdlls.pl" and "filelink.pl"
10. (UNIX) Zip up install files for distribution
11. (NT) Apply an application

Step 6, 7 and 8 may be eliminated if the entire application is going to be installed locally to every workstation. There are also some rare and intelligent applications that can be installed directly to a UNC[19] share. This makes the application image creation process much simpler. To perform this type of installation, step 8 is done first and steps 6 and 7 can be eliminated. When the application is setup it should be installed to the execution UNC share.

Application DLL Resolution

The most significant contribution of the JACAL project to NT software installation is the shared file

15 HKEY_LOCAL_MACHINE\SOFTWARE\...
...Microsoft\Windows\CurrentVersion\RunOnce

16 Or Solitare if you are concerned about network bandwidth

17 Up to 120 minutes if many (25+) machines are running in parallel (These times are based on builds run on our 100Mb ethernet network)

18 According to Microsoft documentation "/m" is the "mandatory option"

19 Universal Naming Convention

conflict resolution system. In Taylor University's computer science labs we run approximately 80 applications on all of our NT workstations. Before we implemented conflicting file resolution we had significant problems with over 2600 files in conflict. The conflict resolving software is composed of two Perl scripts, "mvdlls.pl" and "filelink.pl."

These two scripts detect files which are in common between applications and resolve these conflicts. These scripts are run server side as they make UN*X[20] symlinks which are transparent to the SMB protocol. For all practical purposes, these tools necessitate running SaMBa [21]on a UN*X server.

Applications are stored in separate directories. Each of these directories is the result of a "sysdiff /inf" command. The sysdiff program creates "C" and "$$" directories which represent "C:\" and "C:\winnt\"[22]. Each of these special directories beneath the application directories are recursively searched by "mvdlls.pl" for files that conflict with other applications. When a conflicting file is found,[23] the file is moved to a common _shared_ directory structure. When each conflicting file is placed in the _shared_ structure it is placed in a subdirectory path relative to the C:\ drive. The file is also renamed so that any internal file version[24] is added as a component of the file name as well as the application that the file came from. This convention prevents files from being overwritten by colliding versions in other applications.

Once all conflicts have been detected by "mvdlls.pl" and moved out into the common _shared_ directory, "filelink.pl" is used to select which version of each conflicting file will be part of the final install.

The actual file resolution is achieved by creating a symlink with the original file name which points to the desired version of the file which the administrator has determined should be deployed.

Most file resolution is fairly simple. The largest portion of conflicting files on our site are fonts shared between different Microsoft publishing programs. The key resolutions reconcile different DLL, EXE and OCX versions. Usually in these cases, the most recent version of the executable or library is the best choice.

20 UN*X commonly refers to all variations of Unix
21 SaMBa is a UN*X implementation of Microsoft's Service Message Block (SMB) protocol
22 Actually the directory is the environment variable %SystemRoot%, typically "C:\winnt\"
23 Not just DLLs as the name "mvdlls.pl" suggests
24 File versions are grabbed from internal information from DLLs, EXEs and OCXs

However, this is not always the case.

An example of the resolution process between Visual Studio 97 and Windows Scripting Host would look like this:

Conflicting files:
```
VisualStudio/$$/system32/scrrun.dll
WSH/$$/system32/scrrun.dll
```

Moved to files in the _shared_ directory:
```
_shared_/$$/system32/scrrun.dll.4.0.0.2926.VisualStudio
_shared_/$$/system32/scrrun.dll.3.1.0.2430.WSH
```

Since both files are from Microsoft we create a symlink:
```
_shared_/$$/system32/scrrun.dll
```
Which points to the most recent version:
```
_shared_/$$/system32/scrrun.dll.4.0.0.2926.VisualStudio
```

"filelink.pl" currently has two modes. The first prompts the administrator for file conflicts which have not been resolved. The second mode prompts the administrator to resolve all conflicting files. In the future we will be adding the capability to provide a list of applications for "filelink.pl" to resolve rather than redoing all file resolutions.

Application Installation

The applications or portions of applications that are targeted for the local workstation are stored in separate directories. Each directory is structured like the flat file inf generated by sysdiff. There is an ".inf" file which contains all of the registry and ini modifications. There are also the standard "C" and "$$" directories. However, within these directories are zip files containing the subdirectories and files for the applications. Compressing the applications in this manner reduces the copy time and network bandwidth utilization while adding checksums to the file transfer process.

The installation program copies down the zip file from each of the inf directories. After the archive is on the local disk, files are extracted to their final locations. If the file is read only (i.e. already in use by another program) it is detected from the output of the unzipping program. The file is then placed in the C:\temp directory and "MOVEEX.EXE" is run to place the appropriate registry key into the kernel so NT will move the file on the next reboot. Since applications already have their conflicting files resolved, multiple calls to "MOVEEX.EXE" do not cause any problems.

The applications are all installed in a serial fashion without rebooting. Although at our site we install all applications, it would be very easy to modify the installation serializing script, "zipinst.pl," to check a file that lists applications that should be installed on the workstation. This method results in an incredibly fast and flexible installation of applications with high reliability and complete conflicting file resolution.

Workstation Maintenance

For performing maintenance of workstations that have already been rebuilt we created a Perl script, "wintel.pl", that allows us to execute functions on a large number of workstations in parallel. This script is built to take advantage of Nicolas Deschatrettes's NT Telnet Server. We use it primarily as a way of applying applications and applications fixes in batch. The "wintel.pl" script takes the following parameters: a file that has the list of commands to run on the target machines and a file that has a list of the machines to execute the commands on.

Design Philosophy

Eliminate administrator intervention

Perhaps one of the primary forces driving the computing and information revolution is the belief that repetitive actions performed by people should be handled by Information Technology. It follows that administrators should not be required to manually build multiple workstations that are nearly identical. JACAL is our attempt to eliminate repetition from standard NT and Linux workstation deployment.

Use the best tools for the job

Linux is optimal for manipulating and moving file data around in a heterogeneous environment. Linux is also good at automatically detecting hardware configurations. Auto–detection of hardware is one of the issues that the Bell Labs team [BELL Section 7.4] had trouble performing under NT with their AutoInstall. Linux allows us to easily retrieve and parse information about the hardware. After hardware information is gathered it is used to modify the course of the install process. We currently only use this to detect whether NT should support multiple processors and to detect which video card drivers to use. This auto–detection system could be extended to account for much greater variability in hardware configurations.

Linux does not have tools available to install NT, so we use NT's "SETUP.EXE" for this stage. We use Perl as our primary scripting language, instead of NT batch scripting, because it is very flexible and cross–platform. Eventually we plan to eliminate all the UNIX shell scripts that are still part of JACAL and replace them with Perl scripts.

Use as few development tools as possible

The JACAL project attempts to minimize complexity by limiting the number of development tools. This philosophy keeps JACAL slim and enhances the project's portability, extensibility and maintainability. There is a trade off between using the best tools for the job and using as few tools as possible. We have leaned toward using the best tool but have intentionally designed the system so that the different tools can be replaced or eliminated with a corresponding loss of flexibility or functionality. JACAL currently relies on the following development tools: Perl on both Windows and Linux, UNIX shell scripting, NT batch language, Microsoft's sysdiff utility, Microsoft's ScriptIt tool, limited C.

In keeping with our philosophy of using as few development tools as possible we have designed JACAL without any dependence on NT/2000 Server. In some environments, including ours, Windows NT is only used for desktop workstations. This design philosophy will allow JACAL to be used on other networks like ours and also on networks that contain NT Servers.

Rely on as few network services as possible

Our setup of JACAL requires the following network services: DHCP, NFS, and SMB, and a Linux rsync server. Each one of these services could be eliminated but at the cost of flexibility. DHCP could be eliminated by having a fixed IP address for each build disk. This would require a system to keep track of and give each boot disk a unique IP address.

NFS could be eliminated by storing the JACAL loader and the NT install files on a larger boot media such as a CD–ROM or on a permanent partition on each machine's hard drive. This would have the added benefit of increased build speed but changes to the loader or the install files would be more difficult and time consuming. The NT install files could also be moved from the NFS server to the SMB server.

The application files and install images served from the SMB server could likewise be moved to a local media but the result would be that applications would have to be installed to run locally. This would also limit changes to the applications and would limit the number and size of the applications that can be installed.

Use tools that are flexible and can be automated

We use the criteria of flexibility and automation in evaluating tools for inclusion in the JACAL system.

As an example, one of the major tools that we choose to use for JACAL was Microsoft's sysdiff utility. Although sysdiff itself is not very flexible, it uses a flat file output format that has allowed us to extend the sysdiff system to fit our needs very nicely.

One note of concern in this area is that Microsoft has replaced the sysdiff system in Windows NT 4.0 with VERITAS WinINSTALL LE included in Windows 2000 [WINST]. Sysdiff allowed us to directly access and manipulate the information that was captured during the snapshot process. This has allowed us to eliminate shared file conflicts and to be able to properly maintain and change the application image data as our environment changes. Hopefully we will be able to find a similar method of extracting information from Microsoft's .MSI install files which are created with VERITAS WinINSTALL LE.

Future Goals

Eliminate the JACAL Bootdisk

The most ambitious goal of the project is to move away from the boot disk based model. The JACAL loader would be placed on a small partition on each machine. At the reboot the script would check the network to see if that machine's rebuild flag had been set and then proceed to either re–install the entire setup (specified in a network configuration file) or simply continue to boot the machine. This would increase automation and allow the systems to be locked down by turning off floppy disk boot capability. A bootdisk would only be necessary on a new machine or to update the JACAL boot partition.

Server Controlled Rebuilds

A related goal is a server based command would give the systems administrator power to schedule automatic machine builds. The syntax would be something like:

jacal *date–time <machinename>...*

At the scheduled time the flags for each of the machines in the list would be set for a rebuild and the machines would then be remotely rebooted.

The BeeHive Project

One of the headaches that we have had to deal with in creating application images is that the user registry hive settings must be entered into the global login script. This complicates the application image creation process and slows down user logins. We have started an independent project named BeeHive to create a cross–platform NT/2000 registry editor. This will allow us to edit, repair and optimize roaming profiles from our Linux user space server [BEE].

Rewrite Sysdiff

Another future goal is to write our own system difference tool so that we can simplify the application image creation process and also perform conflict resolution of registry entries and capture incremental changes automatically. We may also write a component which allows us to extract information from Microsoft's new ".MSI" install file format.

FCP (File Cast Protocol)

One project that our team has embarked on is not directly part of the JACAL project but will complement it greatly. This task is to create a cross–platform multi–cast FTP style system so that a large number of workstations can be rebuilt simultaneously with minimal network traffic.

Conclusions

The art of large, automated installation is in a state of rapid change. Both the UNIX and Windows world seem to be converging in the way that large installation and maintenance is carried out. Many UNIX applications are becoming more dependent on local machine resources and as such have become more complicated in their installation process. Microsoft is moving towards a more modularized and protected installation scheme with the Windows Installer under Windows 2000. Both of these trends will certainly continue as UNIX moves toward the desktop and Windows moves toward the server.

JACAL was developed in a very heterogeneous environment and has picked the best parts of each platform in an attempt to create a more manageable network. JACAL is continuing to develop. One of the upcoming challenges will be coexisting with the new Windows Installer system. Another area of growth will be in further development of multi–boot functionality. This is made more important with the growth of viable x86 based operating systems such as BeOS, the BSD series, Solaris, Linux, etc.

The components of JACAL have been released as an Open Source project in order to spur continued growth and to give back to the community which made JACAL possible. Hopefully, JACAL will be a helpful tool for network administrators in many different environments.

JACAL has a permanent home thanks to VA Linux's SourceForge site. At this site you can find the latest documentation and also download the latest distribution of JACAL:

http://jacal.sourceforge.net/

Works Cited:

[ANL] A Comparison of Large-Scale Software Installation Methods on NT and UNIX, Michail Gomberg, Remy Evard and Craig Stacey, Mathematics and Computer Science Division, Argonne National Laboratory

- ➢ http://www-fp.mcs.anl.gov/~stace/Papers/NTLisa1998/ntapps.html
- ➢ http://www-fp.mcs.anl.gov/~stace/Papers/NTLisa1998/ (LISA Presentation)

[BELL] AutoInstall for NT: Complete NT Installation Over the Network, Robert Fulmer and Alex Levine, Lucent Technologies, Bell Labs

- ➢ http://www.usenix.org/publications/library/proceedings/lisa-nt98/fulmer.html

[STATE] State Driven Software Installation for Windows NT, Martin Sjolin, Warburg Dillon Read

- ➢ http://www.usenix.org/events/lisa-nt99/sjolin.html

[HELL] The End of DLL Hell, Rick Anderson. MSDN Library, January 2000

- ➢ http://msdn.microsoft.com/library/techart/dlldanger1.htm

[SID] Windows NT Duplication Problems

- ➢ http://support.microsoft.com/support/kb/articles/q162/0/01.asp
- ➢ http://support.microsoft.com/support/kb/articles/Q183/2/53.asp

[SYS] Sysinternals – Advanced utilities, technical information and source code related to Windows NT/2K

- ➢ http://www.sysinternals.com/

[UNA1] Windows 2000 Professional Automated Deployment Options: An Introduction

- ➢ http://www.microsoft.com/windows2000/library/planning/client/autodeploy.asp

[SMS] Microsoft Announces Availability of Systems Management Server 2.0

- ➢ http://www.microsoft.com/presspass/press/1999/feb99/smspr.asp

[NDT] NT Telnet Server (NDTelnet)

- ➢ http://hem.passagen.se/deschatr/ndtelnet.htm

[WINST] Windows Installer

- ➢ http://www.microsoft.com/windows2000/library/howitworks/management/installer.asp

[WINST] Step-by-Step Guide to Creating Windows Installer Packages

- ➢ http://www.microsoft.com/windows2000/library/planning/management/veritas.asp

[BEE] BeeHive – Cross-platform registry editor

- ➢ http://sourceforge.net/project/?group_id=1987

[RSYNC] rsync is an open sourceutility that provides fast incremental file transfer

- ➢ http://rsync.samba.org/

References:

MS Windows NT Workstation Deployment Guide – Automating Windows NT Setup

- ➢ http://www.microsoft.com/TechNet/winnt/winntas/technote/implemntintegra/gdautset.asp

An Unattended Windows NT Workstation Deployment

- ➢ http://www.microsoft.com/TechNet/winnt/ntwrkstn/technote/ntinstal.asp

Windows NT 4.0 FAQs: Deployment and Unattended Setup Questions

- ➢ http://www.microsoft.com/TechNet/winnt/winntas/technote/troubleshooting/topntqa1.asp

Easier Windows NT Workstation 4.0 Deployment with Disk Image Copying and the MS System Preparation Tool

- ➢ http://www.microsoft.com/TechNet/winnt/ntwrkstn/prodfact/sysprep.asp

Chapter 11 – Windows NT Workstation Unattended Modular Build

- ➢ http://www.microsoft.com/TechNet/winnt/winntas/technote/implemntintegra/manntnet/ntnfch11.asp

Automating Windows NT Setup Deployment Guide Supplement (Sysdiff)

- ➢ http://www.microsoft.com/TechNet/winnt/winntas/technote/implemntintegra/advsysdf.asp

Microsoft's IE Administration Kit

- ➢ http://www.microsoft.com/windows/ieak/

Microsoft's Office 2000 Resource Kit

- ➢ http://www.microsoft.com/office/ork/2000/

ActiveState Perl win32 FAQ

- ➢ http://www.activestate.com/ActivePerl/docs/perlwin32/perlwin32faq.html

The MS ScriptIt Utility

- ➢ http://www.microsoft.com/TechNet/winnt/Winntas/tools/scriptit.asp

Top Ten Windows NT Support Issues

- ➢ http://www.microsoft.com/TechNet/maintain/topsup.asp

Customizing the Windows NT 4.0 Upgrade Process

- ➢ http://www.microsoft.com/TechNet/winnt/winntas/technote/planning/nt4cusup.asp

THE USENIX ASSOCIATION

Since 1975, the USENIX Association has brought together the community of developers, programmers, system administrators, and architects working on the cutting edge of the computing world. USENIX conferences have become the essential meeting grounds for the presentation and discussion of the most advanced information on new developments in all aspects of advanced computing systems. USENIX and its members are dedicated to:

- problem-solving with a practical bias
- fostering innovation and research that works
- communicating rapidly the results of both research and innovation
- providing a neutral forum for the exercise of critical thought and the airing of technical issues

SAGE, the System Administrators Guild

The System Administrators Guild, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession. To join SAGE, you must be a member of USENIX.

Member Benefits:

- Free subscription to *;login:*, the Association's magazine, published eight–ten times a year, featuring technical articles, system administration tips and techniques, practical columns on Perl, Java, Tcl/Tk, and Open Source, book and software reviews, summaries of sessions at USENIX conferences, and Standards Reports from the USENIX representative and others on various ANSI, IEEE, and ISO standards efforts.
- Access to *;login:* on the USENIX Web site.
- Access to papers from the USENIX Conferences and Symposia, starting with 1993, on the USENIX Web site.
- Discounts on registration fees for the annual, multi-topic technical conference, the System Administration Conference (LISA), and the various single-topic symposia addressing topics such as security, Linux, Internet technologies and systems, operating systems, and Windows—as many as twelve technical meetings every year.
- Discounts on the purchase of proceedings and CD-ROMs from USENIX conferences and symposia and other technical publications.
- The right to vote on matters affecting the Association, its bylaws, and election of its directors and officers.
- Savings on a variety of products, books, software, and periodicals: see *http://www.usenix.org/membership/specialdisc.html* for details.

Supporting Members of the USENIX Association:

Earthlink Network
Greenberg News Networks/MedCast Networks
JSB Software Technologies
Lucent Technologies
Macmillan Computer Publishing, USA
Microsoft Research
MKS, Inc.
Motorola Australia Software Centre
Nimrod AS
O'Reilly & Associates Inc.
Performance Computing
Sendmail, Inc.
Server/Workstation Expert
Sun Microsystems, Inc.
Sybase, Inc.
Syntax, Inc.
UUNET Technologies, Inc.
Web Publishing, Inc.

Supporting Members of SAGE:

Deer Run Associates
Electric Lightwave, Inc.
ESM Services, Inc.
GNAC, Inc.
Macmillan Computer Publishing, USA
Mentor Graphics Corp.
Microsoft Research
MindSource Software Engineers
Motorola Australia Software Centre
New Riders Press
O'Reilly & Associates Inc.
Remedy Corporation
RIPE NCC
SysAdmin Magazine
Taos: The Sys Admin Company
Unix Guru Universe

For more information about membership, conferences, or publications,
see *http://www.usenix.org/*
or contact:
USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710 USA.
Phone: 510-528-8649. Fax: 510-548-5738. Email: *office@usenix.org.*

ISBN 1-880446-19-7